TECHNICAL REPORT
NATICK/TR-93/008

AD _____

AN AXISYMMETRIC, TURBULENT FLOW ANALYSIS OF CONTAMINANT INFILTRATION INTO A PRESSURIZED STRUCTURE WITH A FABRIC ENDCAP

by
Struan Robertson
University of Massachusetts
Lowell Research Foundation
Lowell, MA 01854

November 1992

Final Report
January 1990 - December 1990

UNITED STATES ARMY NATICK
RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
NATICK, MASSACHUSETTS 01760-5000

AERO-MECHANICAL ENGINEERING DIRECTORATE

DISCLAIMERS

The findings contained in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of trade names in this report does not constitute an official endorsement or approval of the use of such items.

DESTRUCTION NOTICE

For Classified Documents:

Follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For Unclassified/Limited Distribution Documents:

Destroy by any method that prevents disclosure of contents or reconstruction of the document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1992	Final Jan 1990-Dec 1990

4. TITLE AND SUBTITLE
An Axisymmetric, Turbulent Flow Analysis of Contaminant Infiltration into a Pressurized Structure with a Fabric Endcap

5. FUNDING NUMBERS
DAAK60-92-K-001
PE:62786A
PR:1L162786A427
TA:BO
WU:BOO
AG Code:T/B1380

6. AUTHOR(S)
Struan Robertson, Ph.D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Univ. of Massachusetts Lowell Research Foundation
450 Aiken St.
Lowell, MA 01854

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Natick Research, Development & Engineering Ctr.
Kansas St., ATTN: SATNC-UE
Natick MA 01760-5017

10. SPONSORING MONITORING AGENCY REPORT NUMBER
NATICK/TR-93/008

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The problem of contaminant transport by turbulent flow and its infiltration into a fabric structure pressurized with uncontaminated air is studied for axisymmetric flow. A model for turbulent pipe flow having a concentric, pressurized chamber, with a fabric endcap, placed axially within the pipe is developed. Contaminated air enters the pipe and is transported down the pipe. Some of the contaminated air is transported through the fabric and is dispersed within the pressurized chamber. The effect of forcing uncontaminated air into the chamber on the infiltration and dispersion of contaminated air is studied. The particular formulation that is used in this study is the two-equation model of turbulence. The SIMPLER algorithm and code that were used in earlier laminar flow studies have been modified to incorporate the two-equation model. The fabric is considered to be passive. Comparisons with experimental results show that not all fabrics can be considered as passive because their properties change with the level of contamination.

14. SUBJECT TERMS
AXISYMMETRIC FLOW, CONTAMINATION, NAVIER STOKES EQUATIONS, NUMERICAL MODELS, MODELING, SIMPLER COMPUTER PROGRAM, TURBULENCE, PIPE FLOW, SIMPLER METHOD

15. NUMBER OF PAGES
162

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

This report describes the modeling of airborne contaminant transport and its infiltration into and dispersion within a pressurized chamber with a fabric cover. The airflow is assumed to be turbulent. The work reported here is part of a larger study involving the development of experimental techniques as well as computational tools that can be used to design fabric structures to provide a safe internal environment when the external environment is contaminated. Computationally, the two issues most difficult to deal with are: 1. How much contaminant is deposited on a structure in a contaminated environment and how is it distributed over the structure, and 2. How is the contaminant transported through the fabric that is the interface between the internal and external environments. The computational approach is to deal with the problem as a whole, not to separate the analyses of the external flow, internal flow, and transport through the fabric. The work reported took place from January 1990 to December 1990. It was funded under Contract No. DAAK60-92-K001.

Thanks are due to Mr. John Calligeros who, as director of Natick's Engineering Technology Division, has been most supportive of this work and to Dr. Louis Piscitelle and Dr. Earl Steeves for the many useful discussions we had and for their suggestions. Thanks also to Mr. Gary Vincens and Mr. Kyle Welch for the experimental data they provided. Finally, thanks to Ms. Marcia Lightbody for editing the report and thereby greatly improving it.

An Axisymmetric, Turbulent Flow Analysis of Contaminant Infiltration into a Pressurized Structure with a Fabric Endcap

INTRODUCTION

The problem of contaminant infiltration and dispersal within pressurized fabric structures is highly complex. The level of contamination on the exterior of the fabric could be assumed as a boundary condition. This approach is not satisfactory, since the way airborne contaminants will be distributed over the exterior of a structure is highly dependent on the nature of the external flow, which depends on the geometry of the structure in question. The analysis reported here considers the problem as a whole, not separating the external from the internal flow. Because of this approach, the distribution of contaminant on the exterior of the structure is determined automatically as part of the solution process and need not be specified a priori. The development of the axisymmetric model for the study of the infiltration of airborne contaminants into a fabric–covered chamber, pressurized with uncontaminated air, is discussed in detail. This particular configuration was chosen because an experimental model could be easily developed for it. The external flow is assumed to be turbulent. The Navier–Stokes equations govern the velocity field and the mass transport equation governs the distribution of contaminant. Two equations are used to model the effect of turbulence, an equation for the transport of the turbulent kinetic energy and an equation for the dissipation of turbulent kinetic energy. These two equations are semiempirical. The presence of contaminant is assumed not to affect the velocity field. Thus, the contaminant transport equation is solved after the solution for the flow field is accomplished. The fabric is modeled using an empirically derived relation between pressure drop across the fabric and the velocity through the fabric.

The equations are solved numerically using a control volume, finite difference

method. To permit the use of a relatively coarse mesh, a two–equation model of turbulence is used in conjunction with the Navier–Stokes equations.

The organization of this report is as follows. The Navier–Stokes equations are given along with the associated boundary conditions. The model for the fabric is introduced. Next, the two–equation model for turbulence is introduced with the associated boundary conditions. Turbulence is primarily generated in the very thin boundary layers along walls. In order to avoid the need for very fine grid spacing next to walls, the wall function method, which approximates the velocity profile in the boundary layer, is introduced and discussed in detail. Next, the mass transport equation for the contaminant is introduced with its boundary conditions, followed by discussion of the control volume finite difference technique used to solve the equations. To validate this approach, the problem for turbulent flow through a circular pipe is analyzed. There have been numerous experimental and numerical studies of this problem. The approach is then applied to the problem at hand. The results of various cases and their relation to experimental work conclude the main body of the report. Details concerning the software developed to perform this study are given in the Appendices A–H.

GOVERNING EQUATIONS

Turbulent flow is treated as a statistical process. The instantaneous velocity components are taken as the sum of a mean velocity component and a fluctuating turbulent velocity component. Thus,

$$U = u + \hat{u}$$

$$V = v + \hat{v}$$

where u and v are the mean components, which can be considered as averaged over an interval of time at a point in space for steady–state problems or as an ensemble average at a point in space for transient problems. $\hat{u}$ and $\hat{v}$ are the randomly fluctuating turbulent

velocity components. The mean components are governed by the Navier–Stokes equations. These equations contain products of the fluctuating components, which are called Reynolds stresses. The Reynolds stresses cannot be determined analytically so empirical models are used. These models fall under the name of mixing length models. Their ultimate purpose is to develop a turbulent viscosity coefficient by way of analogy to the laminar viscosity coefficient in order to account for the loss of energy that occurs in the formation of turbulence. There are various mixing length models, one of the most robust being the two–equation or K–e model. The turbulent components are used to define the turbulent kinetic energy and the turbulent energy dissipation function in the two–equation model. This semiempirical approach leads to closure of an otherwise intractable problem and permits the numerical solution of many interesting problems.

Navier–Stokes Equations

The governing equations for the mean velocity components are obtained by averaging the equations for instantaneous velocity components and result in equations directly analogous to those for laminar flow. For axisymmetric problems, the Navier–Stokes equations are, for the radial direction,

$$(1)\qquad \frac{\partial}{\partial t}(\rho u) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho uu) + \frac{\partial}{\partial z}(\rho vu) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\mu_{eff}\frac{\partial u}{\partial r}\right] + \frac{\partial}{\partial z}\left[\mu_{eff}\frac{\partial u}{\partial z}\right] - \frac{\partial p}{\partial r} + S_r$$

and for the axial direction,

$$(2)\qquad \frac{\partial}{\partial t}(\rho v) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho uv) + \frac{\partial}{\partial z}(\rho vv) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\mu_{eff}\frac{\partial v}{\partial r}\right] + \frac{\partial}{\partial z}\left[\mu_{eff}\frac{\partial v}{\partial z}\right] - \frac{\partial p}{\partial z} + S_z$$

where u is the mean radial velocity, v the mean axial velocity, S_r and S_z the source terms which are used to account for loss of momentum (pressure drop) through porous media and $\mu_{eff} = \mu + \mu_t$ is the effective viscosity that includes the effect of turbulent energy dissipation: μ is the laminar viscosity and μ_t is the turbulent viscosity.

The turbulent viscosity is an empirical concept used to account for the work done in the creation of turbulence and its dissipation. This form for the equations does not invoke the continuity equation which is used by the numerical algorithm to develop a pressure equation. The continuity equation for incompressible flow can be written as

$$\frac{1}{r}\frac{\partial}{\partial r}(r\rho u) + \frac{\partial}{\partial z}(\rho v) = 0 \quad . \tag{3}$$

The boundary conditions are normal to a solid boundary

$$\bar{u}\cdot\bar{n} = 0 \quad , \tag{4}$$

tangential to a solid boundary

$$\bar{u}\cdot\bar{t} = 0 \quad , \tag{5}$$

at an inflow boundary

$$\bar{u} \text{ is specified} \quad , \tag{6}$$

and at an outflow boundary

$$\bar{u} \text{ is unspecified} \quad . \tag{7}$$

In the numerical model wall functions are used for velocities at nodes adjacent to walls. This will be discussed in detail later.

Fabric Model

The fabric model is based on the work of Armour and Cannon [1]. The equation is the sum of two terms, one linear in the velocity and the other the square of the velocity. The linear term accounts for viscous drag and the square term for turbulent dissipation. The coefficients in this empirically determined relation depend on the generic characteristics of woven fabric: the void fraction, surface–area–to–volume ratio, effective pore diameter, effective thickness, and the viscosity and density of the fluid. The source terms in the Navier–Stokes equations (1) and (2) are $S_r = 0$ and $S_z = 0$ in the air and are for the fabric

$$S_r = \left[\frac{\alpha\mu A^2}{\epsilon^2} + \frac{\beta\rho}{D\epsilon^2}|\bar{u}|\right]u \tag{8}$$

and

$$S_z = \left[\frac{\alpha\mu A^2}{\epsilon^2} + \frac{\beta\rho}{D\epsilon^2}|\bar{u}|\right]v \tag{9}$$

where the viscous flow resistance coefficient $\alpha = 8.61$ and the turbulent energy dissipation coefficient $\beta = 0.52$. D is the pore diameter, A the surface–area–to–volume ratio, and ϵ is the void fraction. These expressions represent the pressure drop that would occur across the fabric for steady, uniform flow.

Two–Equation Turbulence Model

The Navier–Stokes equations used here are for the mean velocity components. The effect of turbulence is accounted for by adding to the actual viscosity a turbulent viscosity that represents the effect of energy loss in the creation of turbulence. A mechanical energy equation, or equation for the transport of turbulent kinetic energy, is found by taking the momentum equation for the r–direction and multiplying by $\hat{u}$ and the equation for the z–direction and multiplying by $\hat{v}$. After we take the average of each resulting equation and invoke continuity, the resulting equations are added to obtain the desired transport equation (see any text on turbulent flow, e.g. [2]). Half the mean square turbulent velocity is given by,

$$K = \frac{1}{2}(\overline{\hat{u}^2} + \overline{\hat{v}^2}) \tag{10}$$

where $\hat{u}$ and $\hat{v}$ are the randomly fluctuating turbulent velocity components. The turbulent kinetic energy is ρK. The equation for the turbulent kinetic energy is [2,3,4]

$$\frac{\partial}{\partial t}(\rho K) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u K) + \frac{\partial}{\partial z}(\rho v K) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\Gamma_k\frac{\partial K}{\partial r}\right] + \frac{\partial}{\partial z}\left[\Gamma_k\frac{\partial K}{\partial z}\right] + G_k - \rho e \tag{11}$$

where G_k is turbulent kineti · energy generation term given by

$$G_k = \mu_t\{2[u^2_{,r} + v^2_{,z} + (u/r)^2] + (v_{,r} + u_{,z})^2\} \tag{12}$$

where μ_t is the turbulent viscosity which is given by the empirical relation

$$\mu_t = C_d\rho K^2/e \ . \tag{13}$$

In these relations the variable e is the rate of dissipation of turbulent kinetic energy and

(14) $$e = \frac{\mu}{\rho}\left[\overline{\left[\frac{\partial \hat{u}}{\partial r}\right]^2} + \overline{\left[\frac{\partial \hat{v}}{\partial z}\right]^2}\right]$$

The dissipation rate e can be treated like any property that is transported by the flow so the governing transport equation for e is

(15) $$\frac{\partial}{\partial t}(\rho e) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u e) + \frac{\partial}{\partial z}(\rho v e) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\Gamma_e \frac{\partial e}{\partial r}\right] + \frac{\partial}{\partial z}\left[\Gamma_e \frac{\partial e}{\partial z}\right] + C_1 G_k \frac{e}{K} - C_2 \frac{e^2}{K} .$$

The constants C_1, C_2, and C_d are empirical. The diffusion coefficients Γ_k and Γ_e take different forms depending on the nature of the flow [4]. For high Reynolds number flow

(16) $$\Gamma_k = \mu_t/\sigma_k \text{ and } \Gamma_e = \mu_t/\sigma_e$$

and for low Reynolds number flow

(17) $$\Gamma_k = \mu + \mu_t/\sigma_k \text{ and } \Gamma_e = \mu + \mu_t/\sigma_e$$

where σ_k and σ_e are effective turbulent Prandtl numbers for the two transport processes.

The conditions for certain types of boundaries are treated in an ad hoc, but intuitively satisfactory way; on others they are based on experimental results, and on others by semiempirical models called wall functions. The first two are presented in this section while wall functions are dealt with in a separate section.

On symmetry boundaries the normal derivatives vanish

(18) $$\partial K/\partial n = 0 \text{ and } \partial e/\partial n = 0 .$$

At an outlet the normal derivatives vanish

(19) $$\partial K/\partial n = 0 \text{ and } \partial e/\partial n = 0 .$$

At an inlet the turbulent kinetic energy is taken to be some fraction of the average kinetic energy

(20) $$K = c\bar{u}\cdot\bar{u}$$

where, for example, in the case of flow in a circular pipe $c = 0.005$ [5].

The rate of dissipation is taken to be of the form

(21) $$e = C_D K^{3/2}/\ell$$

where ℓ is Prandtl's turbulent mixing length [6,7] and, for example, $\ell = 0.03R$ for a pipe with outer radius R [5].

At a wall, it is recommended that [4]

(22) $$\partial K/\partial n = 0 \ .$$

This is reasonable since there is no energy flux through a wall. The rate of dissipation e at a wall is treated with wall functions.

Wall Functions

In theory, it is possible to numerically model the flow field next to the wall without resorting to any special methods. Practically, however, it is not desirable to treat boundary layers in great detail because this would require a very fine mesh indeed. This is because the nature of the flow changes so dramatically from laminar adjacent to the wall, to transition and then to turbulent in a very short distance. Jones and Launder [6], for example, used almost 100 nodes in the boundary layer to model turbulent flow through a pipe. In order to alleviate the need for such refinement, when the full details of the boundary layer are not required, a semiempirical method called the wall function technique was developed [4,5,7] and applied successfully to several interesting problems. The method makes certain assumptions about the nature of the flow in the boundary layer and is thus not strictly applicable to situations where those assumptions are violated. The concept could, however, be extended. In addition to the assumptions about the flow in the boundary layer, empirical results are used to determine appropriate constants. Thus, the method is semiempirical.

The process is based on the premise that because the boundary layer is so thin in the neighborhood of the wall, the velocity profile for many walls will not be significantly different from the profile near the wall for turbulent pipe flow. The problem of turbulent

pipe flow is well understood and there is a well-established body of theoretical and experimental work for it. Using these results an empirical model of the behavior of flow in the boundary layer can be simply quantified, thereby obviating the need to model the boundary layer in detail. The use of this model for flow near the wall yields appropriate boundary conditions and source terms for the momentum, energy and dissipation equations.

In developing the model for flow near the wall, assumptions need to be made. The first assumption is that flow in the boundary layer is planar, or nearly so. The second assumption is that the flow can be modeled using the assumptions of Couette flow. This classical viscous flow problem models the flow between two parallel plates, one fixed and the other moving with constant velocity. The next assumption is that Prandtl's mixing length hypothesis holds[4,7,8] for points far enough away from the wall such that they are in the fully turbulent region:

$$\ell = \kappa Y \quad , \tag{23}$$

where ℓ is the mixing length κ is von Karman's constant and Y is the normal distance from the wall to a point in the turbulent part of the boundary layer. The effective viscosity is related to the gradient of the tangential velocity by

$$\mu_{eff} = \mu + \rho \ell^2 \left| \frac{dU}{dY} \right| \quad , \tag{24}$$

where U is the tangential velocity. Further, for high velocity gradients μ is small compared to the second term, so

$$\mu_{eff} = \rho \kappa^2 Y^2 \left| \frac{dU}{dY} \right| \quad . \tag{25}$$

It is further assumed that the pressure gradient in the boundary layer is negligible and that there is negligible mass flux. Also, physical properties are assumed to be constant and, as is usually done [2,8] in developing the law of the wall, the flow is taken to be in the positive direction. With these conditions the momentum equations show that

$$\tau = \tau_w \quad , \tag{26}$$

where τ is the shear stress in the boundary layer and τ_w is the shear stress on the wall, i.e., the shear stress is uniform in the boundary layer. Since,

$$\tau = \mu_{eff} dU/dY \quad , \tag{27}$$

using eqs. (25) and (26) yields

$$\tau_w = \rho\kappa^2 Y^2 \left|\frac{dU}{dY}\right| \frac{dU}{dY} \quad . \tag{28}$$

This can be written as

$$\frac{dU}{dY} = \frac{\sqrt{\tau_w/\rho}}{\kappa Y} \tag{29}$$

which can be integrated directly. However, it is usual to introduce some nondimensional variables first. Thus, define

$$Y^* = \sqrt{\tau_w \rho}\, Y/\mu \quad , \tag{30}$$

$$U_f = \sqrt{\tau_w/\rho} \quad , \tag{31}$$

which is called the friction velocity, and

$$U^* = U/U_f \quad . \tag{32}$$

Using eqs. (30)–(32) in eq. (29) yields

$$\frac{dU^*}{dY^*} = \frac{1}{\kappa Y^*} \quad , \tag{33}$$

which can be integrated to give

$$U^* = \kappa^{-1} \ln(EY^*) \quad , \tag{34}$$

where E is a constant of integration. Von Karman's constant κ and the "wall roughness" constant E are empirically determined. Eq. (34) is called the logarithmic law of the wall. Substitution of Eqs. (30) – (32) into eq. (34) shows that the wall stress appears on both sides of the equation, thus giving a transcendental equation for τ_w [8]. If all the above assumptions apply, then the the rate of turbulent energy generation and the rate of turbulent energy dissipation in the boundary layer are in balance. This balance has been demonstrated experimentally [2,8] and implies for the case of uniform shear stress in the boundary layer that

(35) $$\tau_w/\rho = C_D^{1/2}K = \text{constant} \ .$$

Therefore, $Y^* = \rho C_D^{1/4}K^{1/2}Y/\mu$. Since $(\tau_w/\rho)^{1/2} = (\tau_w/\rho)/(\tau_w/\rho)^{1/2}$ eqs. (31), (32), (34), and (35) combine to give

$$\tau_w = \rho C_D^{1/4}K^{1/2}\kappa U/\ln(EY^*) \ .$$

This wall law is used if $11.5 \leq Y^* \leq 30.0$ and Newton's law with laminar viscosity is used if $Y^* \leq 11.5$. The lower limit is obtained by the requirement that the velocity in the laminar (linear) sublayer match the velocity in the logarithmic layer. The upper limit is obtained from the requirement that the velocity in the logarithmic layer match the power law profile for the turbulent core, obtained from empirical results for pipe flow, to a high degree of accuracy. These limits can vary somewhat depending on the magnitude of the Reynolds number. The values chosen, however, cover a wide range of flows and have been recommended by Launder and Spalding [4,7]. In summary, for a point p in the boundary layer

(36a) $$\tau_w = \rho C_D^{1/4}K^{1/2}\kappa U_p/\ln(EY^*) \quad 11.5 \leq Y^* \leq 30.0 \ ,$$

(36b) $$\tau_w = \mu(dU/dY)_p \quad Y^* \leq 11.5 \ ,$$

with

(37) $$Y^* = \rho C_D^{1/4}K^{1/2}Y_p/\mu \ .$$

In a numerical model, therefore, it is desirable to have the first node adjacent to the wall such that $11.5 \leq Y^* \leq 30.0$, if the flow is indeed turbulent. This requires trial and error since K is not known a priori.

In the turbulent energy equation (11) the source term is comprised of two parts, $S_K = G_K + \rho e$. From eq. (13) $e = C_D\rho K^2/\mu_t$. The fundamental relation for G_K in terms of shear stress is $G_K = \tau_w(dU/dY)_p$. Assuming the velocity gradient is large in the boundary layer gives $\mu_{eff} = \mu_t$. As a result $\mu_t = \tau_w/(dU/dY)_p$ can be used in the expression for e . Thus the source term for the turbulent kinetic energy equation (11) for points in the boundary layer is

(38) $$S_K = (\tau_w - C_D\rho^2K^2/\tau_w)(dU/dY)_p \ .$$

For the dissipation equation, the assumption that $e \alpha K^{3/2}/\ell$ leads to the expression

(39) $$e_p = C_D^{3/4}K_p^{3/2}/\kappa Y_p \ ,$$

for points in the turbulent part of the boundary layer. Thus, instead of a boundary condition being specified for e on a wall, eq. (39) is used to fix the value of e at a point near the wall.

These results will be incorporated into the numerical model later on.

Mass Transport Equation

As is well known, different quantities have different rates of diffusion. In turbulent flow the process of diffusion is augmented and often overwhelmed by turbulent mixing. Thus, the usual practice to account for this effect has been to add a turbulent diffusion term to the molecular diffusion term. For example, as was stated earlier, $\tau = (\mu + \mu_t)\partial u/\partial y$ where μ_t accounts for the turbulent contribution to viscosity. This is often written as $\mu_t = \rho\epsilon_m$ where ϵ_m is called the eddy viscosity or eddy diffusivity of momentum. By Reynolds analogy, it is expected that the diffusion of other quantities such as heat, mass, etc. would behave similarly [2]. In fact, just such an assumption is made in the turbulent kinetic energy and kinetic energy dissipation equations. For mass diffusion in laminar flow, Fick's Law states that $M = -\eta\partial C/\partial y$ where M is the mass flux per unit area, C is the mass concentration and η is the diffusion coefficient. By analogy, for turbulent flow this would be

$$M = -\eta_{eff}\partial C/\partial y = -(\eta + \epsilon_d)\partial C/\partial y \ .$$

The ratio of eddy diffusivity of momentum to eddy mass diffusivity is called the turbulent Schmidt number

(40) $$Sc_t = \epsilon_m/\epsilon_d \ .$$

When $Sc_t = 1.0$ this is called Reynolds analogy. Launder and Spalding [7] recommend using $Sc_t = 0.7$. Whatever value is picked, since $\epsilon_d = \epsilon_m/Sc_t = \mu_t/\rho Sc_t$,

the effective mass diffusion coefficient is given by

$$\eta_{\text{eff}} = \eta + \mu_t/\rho \text{Sc}_t \quad . \tag{41}$$

More details can be found in Reynolds' book [2]. For the fabric, the laminar diffusion coefficient η is found by using the Tsai–Halpin equation [9] as follows:

$$\eta_{\text{fabric}} = (1 + \lambda V_f)\eta_{\text{air}}/(1 - \lambda V_f) \, , \tag{42}$$

where

$$\lambda = (\eta_{\text{fiber}} - \eta_{\text{air}})/(\eta_{\text{fiber}} + \eta_{\text{air}})$$

and $V_f = 1 - \epsilon$ is the volume fraction of fiber.

More details can be found in Reynolds book [2].

The governing equation for mass transport is

$$\frac{\partial(C)}{\partial t} + \frac{1}{r}\frac{\partial(ruC)}{\partial r} + \frac{\partial(vC)}{\partial z} = \frac{1}{r}\frac{\partial}{\partial r}\left[r\eta_{\text{eff}}\frac{\partial C}{\partial r}\right] + \frac{\partial}{\partial z}\left[\eta_{\text{eff}}\frac{\partial C}{\partial z}\right] \quad . \tag{43}$$

The boundary conditions are either

$$C \text{ specified} \tag{44}$$

or

$$\partial C/\partial n = 0 \tag{45}$$

at a wall or a symmetry boundary. In this case wall functions are not used since these are used only to obtain a representation of the shear stress at the wall.

In summary the governing equations are the Navier–Stokes equations for the velocity components u and v.

$$\frac{\partial}{\partial t}(\rho u) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho uu) + \frac{\partial}{\partial z}(\rho vu) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\mu_{\text{eff}}\frac{\partial u}{\partial r}\right] + \frac{\partial}{\partial z}\left[\mu_{\text{eff}}\frac{\partial u}{\partial z}\right]$$
$$-\frac{\partial p}{\partial r} + S_r$$
$$\frac{\partial}{\partial t}(\rho v) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho uv) + \frac{\partial}{\partial z}(\rho vv) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\mu_{\text{eff}}\frac{\partial v}{\partial r}\right] + \frac{\partial}{\partial z}\left[\mu_{\text{eff}}\frac{\partial v}{\partial z}\right]$$
$$-\frac{\partial p}{\partial z} + S_z \, ,$$

the continuity equation for incompressible flow,

$$\frac{1}{r}\frac{\partial}{\partial r}(r\rho u) + \frac{\partial}{\partial z}(\rho v) = 0 \ ,$$

the equations for turbulent kinetic energy and turbulent dissipation,

$$\frac{\partial}{\partial t}(\rho K) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u K) + \frac{\partial}{\partial z}(\rho v K) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\Gamma_k\frac{\partial K}{\partial r}\right] + \frac{\partial}{\partial z}\left[\Gamma_k\frac{\partial K}{\partial z}\right] + G_k - \rho e$$

$$\frac{\partial}{\partial t}(\rho e) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u e) + \frac{\partial}{\partial z}(\rho v e) = \frac{1}{r}\frac{\partial}{\partial r}\left[r\Gamma_e\frac{\partial e}{\partial r}\right] + \frac{\partial}{\partial z}\left[\Gamma_e\frac{\partial e}{\partial z}\right] + C_1 G_k\frac{e}{K} - C_2\frac{e^2}{K} \ ,$$

and the mass transport equation for the contaminant

$$\frac{\partial(C)}{\partial t} + \frac{1}{r}\frac{\partial(ruC)}{\partial r} + \frac{\partial(vC)}{\partial z} = \frac{1}{r}\frac{\partial}{\partial r}\left[r\eta_{eff}\frac{\partial C}{\partial r}\right] + \frac{\partial}{\partial z}\left[\eta_{eff}\frac{\partial C}{\partial z}\right] \ .$$

Generic Field Equation

Examination of the governing equations shows, with the exception of the continuity equation, they are of the general form

$$\nabla\cdot(\rho\phi\vec{u}) = \nabla\cdot(\Gamma\nabla\phi) + S \ . \tag{46}$$

This general form was used in the development of the SIMPLER algorithm [10] which is used to perform the numerical analysis discussed in the next section. The continuity equation is used in the SIMPLER approach to obtain both a pressure and a pressure correction equation the details of which are given in [10]. The pressure equation and the pressure correction equation are buried in the code and are "invisible" to the user.

MODELING CONSIDERATIONS

Incorporation of Turbulence into the SIMPLER Algorithm

The SIMPLER algorithm as developed by Patankar [10] for laminar flows is modified to accommodate the two–equation model of turbulence. Both the turbulent kinetic energy equation (11) and the dissipation of turbulent energy equation (15) have the form of the generic equation (46). The code that is used is described in an earlier report by Robertson [11]. The code is designed to solve the momentum equations, the pressure equation, and the pressure correction equation, plus any number of additional equations that have the

form of the generic equation. The pressure and pressure correction equation are obtained from the continuity equation, the details of which can be found in [10]. These are hidden from the user in the code. Additional equations can be solved along with the flow equations if they are coupled to the velocity field, as is the case with the two equations used to model turbulence, or they can be solved separately after the flow field is determined, as is the case with the mass transport equation for the case of low concentrations. Some modification of the code is required to achieve this. The first modification is to force the code to solve the two equations used to describe turbulence iteratively along with the flow equations. The second modification is to incorporate the wall function technique for modeling boundary layer effects. No special changes to the code are required to solve the mass transport equation.

The code is broken into two main parts; the solution engine and the driver. The solution engine is, for practical purposes, a black box and once set up needs not be modified. The driver is a set of user–developed routines that describe the numerical problem to be solved and control the solution scheme. The driver communicates to the engine through COMMON blocks. Originally, to save storage the code made extensive use of high speed storage for the various arrays used by the code. The use of high speed storage has been completely eliminated and the code now runs in core, which has sped up the solution process considerably. To incorporate turbulence only two changes were made to the engine. These are CALL statements to ENTRY locations in the driver for computing the generation term G_K and to use the wall functions describing boundary layer effects. The required numerical expressions for the wall functions and G_K are developed here. The changes to the code are detailed in Appendix A. The driver has the same general form as reported earlier [11], but to incorporate turbulence effects several major additions were required. The numerical expressions that must be added are given here and the coding details appear in Appendix B. Note that in the code the axial

axial direction is x and the radial direction is y while the axial velocity is u and the radial velocity is v.

The SIMPLER algorithm uses a staggered grid approach, as shown in Figure 1. Main grid points are defined by the user. Around each grid point the code sets up a control volume. All scalar variables and all properties are located at these main grid points, i.e., temperature, pressure, concentration, density, viscosity, etc. Velocity components are located on the faces normal to their flow direction. This leads to two additional sets of control volumes, one for each component. This rather awkward construction has important consequences for the stability and convergence of the algorithm [10]. So what is lost in terms of convenience is gained in performance. This grid staggering requires a significant amount of interpolation when moving variables and properties back and forth from the momentum to scalar equations. Additionally, source terms are not always incorporated in what would appear to be the "obvious" way, again, to improve convergence. The changes required for each equation will be taken in turn starting with the velocity equations. The solution scheme is iterative so values from the previous step are used in the present step.

Velocity Equations

The turbulent viscosity μ_t is given by eq. (13). In the code this is defined in the driver at ENTRY DENSE and has the form,

(47) AMUT(I,J) = CD*RHO*TKE(I,J)**2/TED(I,J) ,

where AMUT = μ_t , TKE = K , and TED = e . The effective viscosity μ_{eff} is defined at ENTRY GAMSOR for the velocity components and has the form

(48) GAM(I,J) = AMU + AMUT(I,J) ,

where AMU = μ and GAM = μ_{eff}. The effect of walls on the velocity equations requires modification to the coefficients in the finite difference expressions. The effect of a wall on the "north" side of a control volume will be detailed.

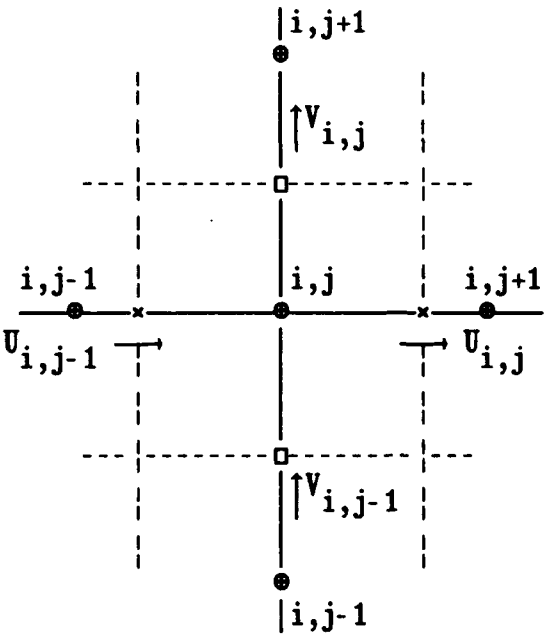


Figure 1a. The locations where u and v are evaluated, ● main grid nodes, × u-nodes, and □ v-nodes.

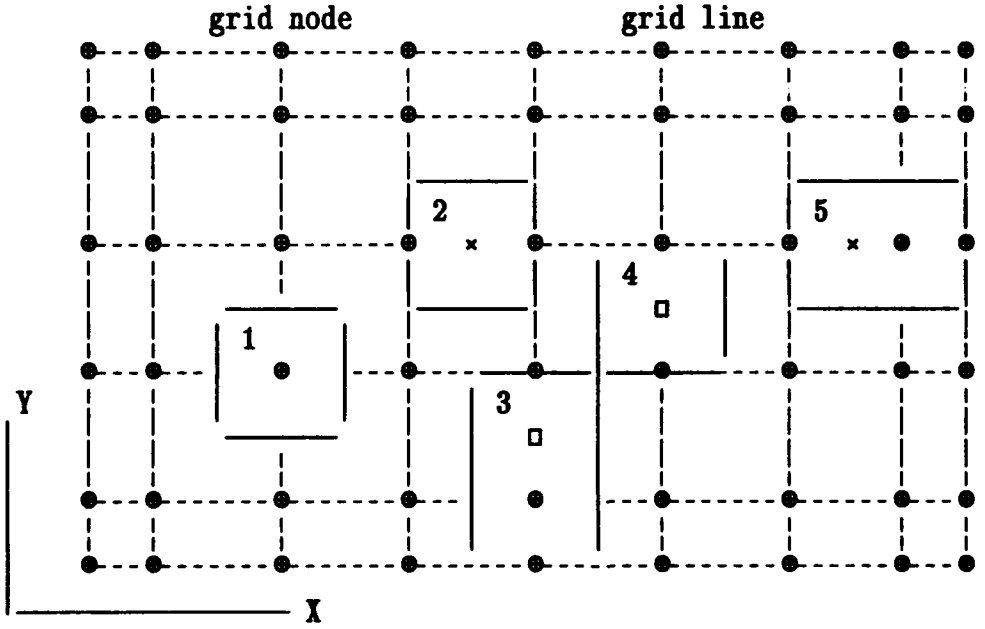


Figure 1b. Typical control volumes; CV 1 scalars, CV 2 u, CV 3 v boundary, CV 4 v, and CV 5 u boundary.

Figure 2 shows the control volume for the u – velocity component at a wall on the north face. The governing equations all have the form of the generic equation [46] and so their finite difference forms will be similar. For the u–velocity component the form of the finite difference equation [10] is

(49) $$a_p u_p = a_e u_e + a_w u_w + a_n u_n + a_s u_s + (\rho_p u_p^o/\Delta t + S_c)\Delta x \Delta y \ ,$$

where the a's are the finite difference coefficients and $u_p = u_{ij}$ is the unknown value. S_c is the constant part of the source term and S_p is the linear part. The a's are essentially force contributions averaged over control volume faces and

(50) $$a_p = a_e + a_w + a_n + a_s + (\rho_p/\Delta t - S_p)\Delta x \Delta y$$

The expression that is obtained for a_n in developing the finite difference equations for laminar flow is

(51) $$a_n = \mu \, \Delta A/\Delta y \ ,$$

where ΔA is the area of the north face and Δy is the distance from the wall to node p. Even though $u_n = 0$ at the wall there is a contribution to the coefficient of u_p that accounts for the drag due to the wall. This coefficient must be modified to account for turbulent boundary layer flow. Referring to eq. (37),

(52) $$Y^* = \rho C_D^{1/4} K^{1/2} \, \Delta y/\mu \ ,$$

and referring to eqs. (36a & b),

(53a) $$\tau_w = [\kappa \rho C_D^{1/4} K^{1/2}/\ln(EY^*)]u_p \quad 11.5 \le Y^* \le 30 \ ,$$

(53b) $$\tau_w = \mu u_p/\Delta y \quad Y^* \le 11.5 \ .$$

Thus a_n becomes

(54) $$a_n = \tau_w \, \Delta A/u_p \ .$$

By using equations (52) to (54) the appropriate expressions to use in the code can be obtained.

The first application is to turbulent flow in the pipe so the use of the above methodology will be illustrated with reference to the driver for this problem in

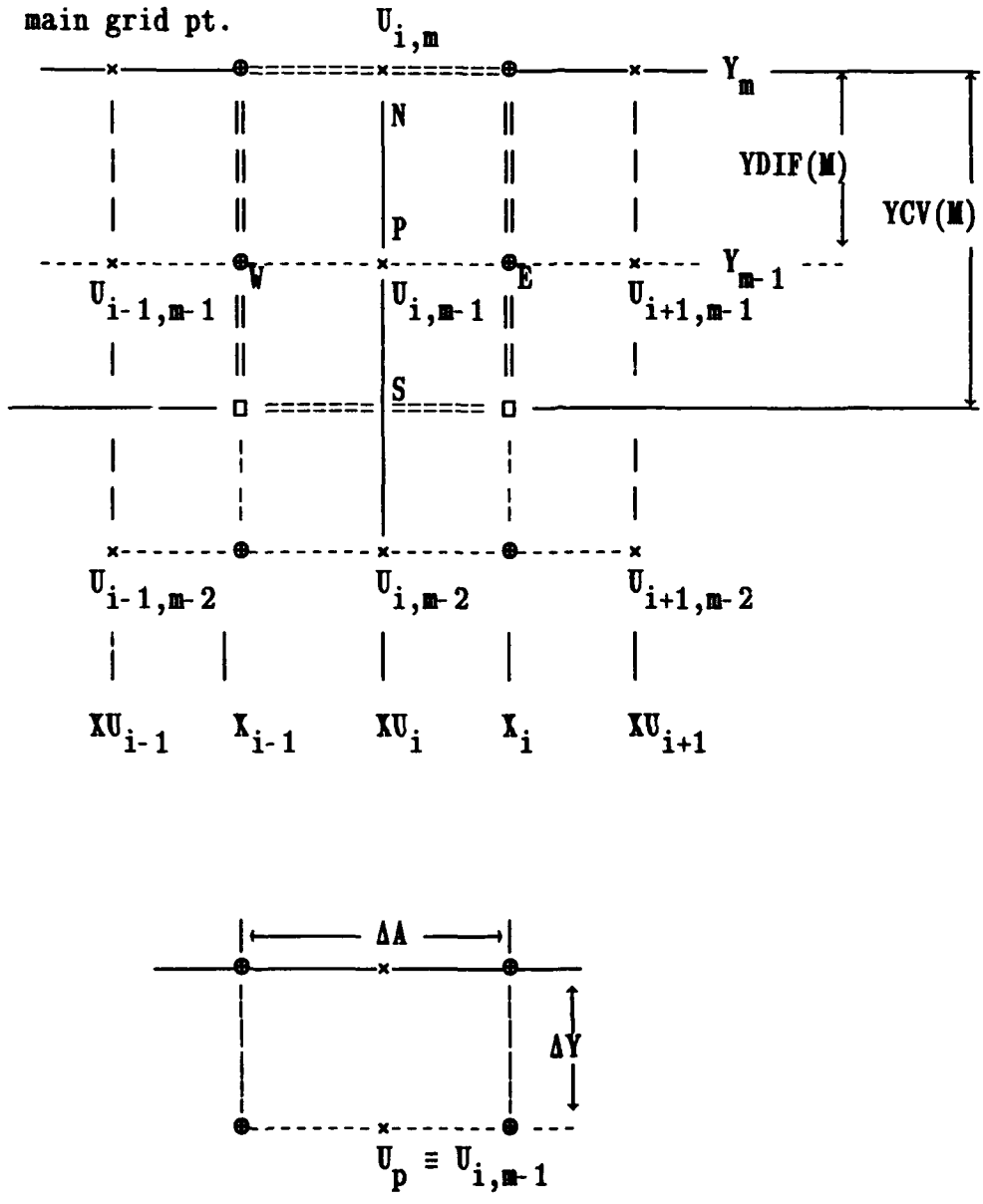


Figure 2. U - control volume on a north boundary or wall.

Appendix B. The ratio of wall stress to velocity is calculated at each point along the wall. It is called TAUN(I) and is calculated in the driver at ENTRY BOUND as follows: Let

DIST = ΔY

CD25 = $C_D^{1/4}$

RHOCON = ρ

TKE = K

PLUN = Y^*

TAUN = τ_w/u_p

CAPPA = κ

ECON = K

then

```
DIST = Y(M1) – Y(M2)
CT = CD25*DIST
RK = RHOCON*SQRT(ABS(TKE(I,M2)))
PLUN(I) = RK*CT/AMU
IF (PLUN(I) .GT. 11.5) THEN
  EPLUN = ECON*PLUN(I)
  TAUN(I) = CAPPA*RK*CD25/DLOG(EPLUN)
ELSE
  TAUN(I) = AMU/DIST
ENDIF .
```

As mentioned earlier, it is desirable to have Y^* (PLUN(I)) between 11.5 and 30 but since it depends on K (TKE(I,J)) trial and error is necessary to have the first node next to the wall properly located. Having calculated TAUN at all the grid points along the wall, we can apply the wall function method to the various equations. The value of TAUN is calculated at the main grid points and must be interpolated to find the value at the

velocity grid points. The value of a_n for the u – equation (49) is calculated in the driver at ENTRY WALL as follows,

AJP(I,M2) = (TAUN(I–1)*FXM(I) + TAUN(I)*FX(I))*XDIF(I)*Y(M1) .

where FXM and FX are interpolants derived in the main part of the code. They are defined in Appendix A. Since Y(M1) is the radius of the pipe, XDIF(I)*Y(M1) is ΔA.

Turbulent Energy

The diffusion coefficient for the turbulent energy equation is given by eq. (17) and is calculated in the driver at ENTRY GAMSOR as

$$\Gamma_k = \text{GAM(I,J)} = \text{AMU} + \text{AMUT(I,J)/PRTKE} ,$$

where PRTKE = σ_k .

It is necessary to determine the turbulent energy generation function G_k given by eq. (12) by appropriately evaluating the partial derivatives interpolated to the main grid points. Referring to Figure 3, the following calculation is performed at ENTRY GRADIENT in the the driver.

$u_{,x}$ = UPX = (U(I+1,J) – U(I,J))/XCV(I)

$v_{,r}$ = VPY = (V(I,J+1) – V(I,J))/YCV(J)

v/r = VOR = 0.5*(V(I,J+1) + V(I,J))/Y(J)

FAC1 = (YV(J+1) – Y(J))/YDIF(J+1)

FAC2 = (YV(J) – Y(J–1))/YDIF(J)

UNEC = U(I,J) + (U(I,J+1) – U(I,J))*FAC1

USEC = U(I,J–1) + (U(I,J) – U(I,J–1))*FAC2

UNWC = U(I+1,J) + (U(I+1,J+1) – U(I+1,J))*FAC1

USWC = U(I+1,J–1) + (U(I+1,J) – U(I+1,J–1))*FAC2

$u_{,r}$ = UPY = 0.5*(UNEC – USEC + UNWC – USWC)/YCV(J)

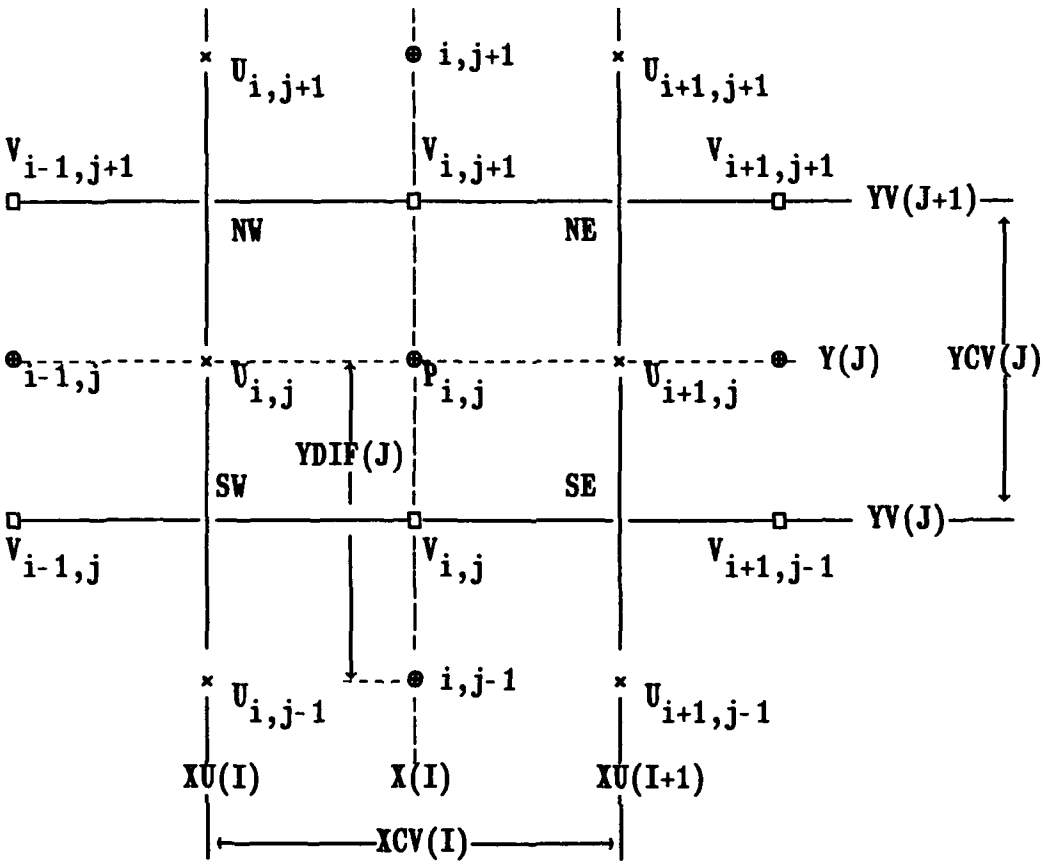


Figure 3. Velocity components used to calculate partial derivatives at the main grid point $P_{i,j}$.

$v_{,x}$ = VPX = 0.5*(V(I+1,J+1) – V(I–1,J+1) + V(I+1,J) – V(I–1,J))
/(X(I+1) – X(I–1))

G_k = GK(I,J) = AMUT(I,J)*(2.*(UPX*UPX + VPX*VPX + VOR*VOR)
+ (UPY + VPX)**2) .

UNEC, USED, UNWC, and USWC are the interpolated values of U at the corners of the control volume. The source term for the turbulent energy equation (11) is $S_k = G_k - \rho e$. This is rearranged, according to the procedure used by Pun and Spalding [5], in order to improve convergence as follows,

(55a) $$S_k = SC_k + SP_k*K$$

where

(55b) $$SC_k = 1.5G_k + (C_2 - 1)\rho e$$

and

(55c) $$SP_k = -(0.5G_k + C_2\rho e)/K \ .$$

This is calculated at ENTRY GAMSOR in the driver where

$$CON(I,J) = SC_k$$

is the constant part of the source term and

$$AP(I,J) = SP_k$$

is the linear part. The effect of the wall on the turbulent energy equation is taken into account through modification of the source terms as per eq. (38). This will be examined for grid nodes next to a north wall. Since $\tau_w = \mu_{eff}\partial u/\partial y$ eq. (38) is of the form $S_k = \mu_{eff}(\partial u/\partial y)^2 - C_d\rho^2K^2/\mu_t$ and since $\tau_w = u_p$*TAUN(I) , the following procedure is used

(56a) $$SC_k = \tau_w \partial u/\partial y \ ,$$

(56b) $$SP_k = -C_D(\rho^2/\tau_w)K\partial u/\partial y$$

which, at ENTRY GAMSOR, takes the form

u_p = UP = 0.5*(U(I+1,M2) + U(I,M2))

$(\partial u/\partial y)_p$ = UPY = UP/YDIF(M1)

τ_w = TAUNW(I) = TAUN(I)*UP

SC_k = CON(I,J) = TAUNW(I)*UPY

SP_k = AP(I,M2) = –CD*RHOCON**2*TKE(I,M2)*UPY/TAUNW(I) .

No special boundary conditions are used.

Energy Dissipation

The diffusion coefficient for the energy dissipation is given by eq. (17). This is calculated in the driver at ENTRY GAMSOR by

$$\Gamma_e = \text{GAM(I,J)} = \text{AMU} + \text{AMUT(I,J)/PRTED} ,$$

where PRTED = σ_e .

From eq. (15), the source term for the turbulent energy dissipation equation is $S_e = C_1 G_k e/K - C_2 e^2/K$. Again, following the treatment of Pun and Spalding [5], this is rewritten as

(57a) $$S_e = SC_e + SP_e * e$$

where

(57b) $$SC_e = C_1 G_k e/K + (C_2 - 1)\rho e^2/K$$

and

(57c) $$SP_e = -(2C_2 - 1)\rho e/K \ .$$

This is calculated at ENTRY GAMSOR by the following statements

e/K = TEOK = TED(I,J)/TKE(I,J)

ρe = RHOTED = RHOCON*TED(I,J)

SC_e = CON(I,J) = (C1*GK(I,J) + C2M*RHOTED)*TEOK

SP_e = AP(I,J) = –TC2M*RHOCON*TEOK .

As mentioned earlier, the effect of a wall on this equation is accounted for by forcing the value of e for nodes next to a wall to be that defined by eq. (39). This is done in the driver at ENTRY GAMSOR by the following two statements,

$$SC_e = CON(I,M2) = CD75*ABS(TKE(I,M2)**1.5)*1.E30$$
$$/(CAPPA*YDIF(M1))$$

and

$$SP_e = AP(I,J) = -1.E30 \ .$$

Concentration Equation

The diffusion coefficient for the concentration (mass transport) equation is given by eq. (41). The first numerical example does not consider mass transport but the second one does. The concentration is assumed to be so low that the velocity equations are not significantly affected. Thus, the concentration equation (43) is solved after the velocity field is determined. The code uses the form of the generic equation (46) for all variables that obey the mass transport equation. It is, therefore, necessary to multiply eq. (43) through by ρ in order to put it in the correct form. A separate driver is developed for this equation that merely reads in the velocity field and solves only the concentration equation. The diffusion coefficient, suitably modified, is calculated at ENTRY GAMSOR by

$$\rho\eta_{eff} = GAM(I,J) = RHOCON*ETA + AMUT(I,J)/PRCON \ ,$$

where $PRCON = Sc_t$.

There are no source terms in this case and walls are assumed to have no special effect on the concentration.

NUMERICAL MODELS

The values for the various constants in the two–equation model for turbulence are those used by Launder and Spalding [7]. These and the values for the von Karman constant κ and the wall roughness constant E in the wall function model are, C_1=1.44, C_2=1.92, C_d=0.09, σ_k=1.00, σ_e=1.30, Sc_t=0.70, κ = 0.40, and E=9.0.

Flow Through a Circular Pipe

This problem is used to determine the effectiveness of the code used here by comparing results with those of other workers. The first problem examined is one considered by Jones and Launder [6]. They did not use wall functions but, instead, used a very fine mesh through the boundary layer (on the order of 100 nodes). They quite effectively modeled the experimental results of others. The first two cases assume a power law distribution that fits the experimental data for fully developed flow at the inlet and the same distribution for the initial velocity field in the pipe. At first glance this would appear to be trivial but, it should be noted, the pressure field is unknown so, as the analysis iterates to a solution, the velocity field will in fact change from iteration to iteration until the pressure field stabilizes. If, when the pressure field stabilizes, the solution has converged to fully developed flow along the length of the pipe, then the algorithm for the representation of turbulence can be considered satisfactory. The last case looks at developing flow and is compared with numerical results obtained by others.

Case 1.

This first case is for a Reynolds number of Re = 500,000 based on the maximum velocity. The pipe has a length of 8.0 and a radius of 0.1. The density is $\rho = 1.0$, the viscosity $\mu = 0.0001$. The inlet velocity distribution obeys a power law giving a distribution close to the experimentally obtained distribution of Laufer [2,6,12]. This is expressed by

$$u_{inlet} = u_{max}(1 - r/r_{max})^{1/9}$$

where $u_{max} = 250$. The domain has 92 axial grid points and 42 radial grid points. The spacing is uniform axially and is nonuniform radially with a power law distribution that becomes finer closer to the wall since the boundary layer is concentrated within 1.0% of the pipe's outer radius. Fig. 4 shows the good agreement between the model and experiment in

the fully developed region near the exit from the pipe. Fig. 5 shows the computed velocity distribution near the pipe's exit. Clearly, as would be expected, it is not parabolic.

Case 2.

In this case Reynolds number is Re = 50,000 based on the maximum pipe velocity. The problem specifications are the same as for Case 1 except that the inlet velocity distribution is given by

$$u_{inlet} = u_{max}(1 - r/r_{max})^{1/8}$$

where $u_{max} = 25$. No experimental results were available for this case but Reynolds [2] outlines a procedure for determining an empirically validated power law representation of the velocity, which yields for the fully developed flow region,

$$u = u_{max}(1 - r/r_{max})^{1/8.229} .$$

Fig. 6 shows that the agreement between the code and this expression is reasonably good but slightly worse than for Case 1. This can be attributed to the fact that the wall function method works best for high Reynolds numbers.

Case 3.

This last case considers a problem solved by Pun and Spalding [5] for a Reynolds number of Re = 100,000. The inlet velocity is taken to be uniform with the value $u_{in} =$ 50. The density is $\rho = 1.0$ and the viscosity is $\mu = 0.0001$. The pipe's length is 2.0 and its radius is 0.1 . The mesh is crude being only 10 X 10 with equal spacing in both the axial and radial directions. Because the model is short, the flow is still developing upon exit. The mesh is the same as used by Pun and Spalding. This example was chosen because it permitted the comparison of two computer codes that use slightly different solution strategies as well as having been programmed by different individuals. Fig. 7 shows the very close agreement between the two models.

The output for Cases 1 and 2 appear in Appendix B.

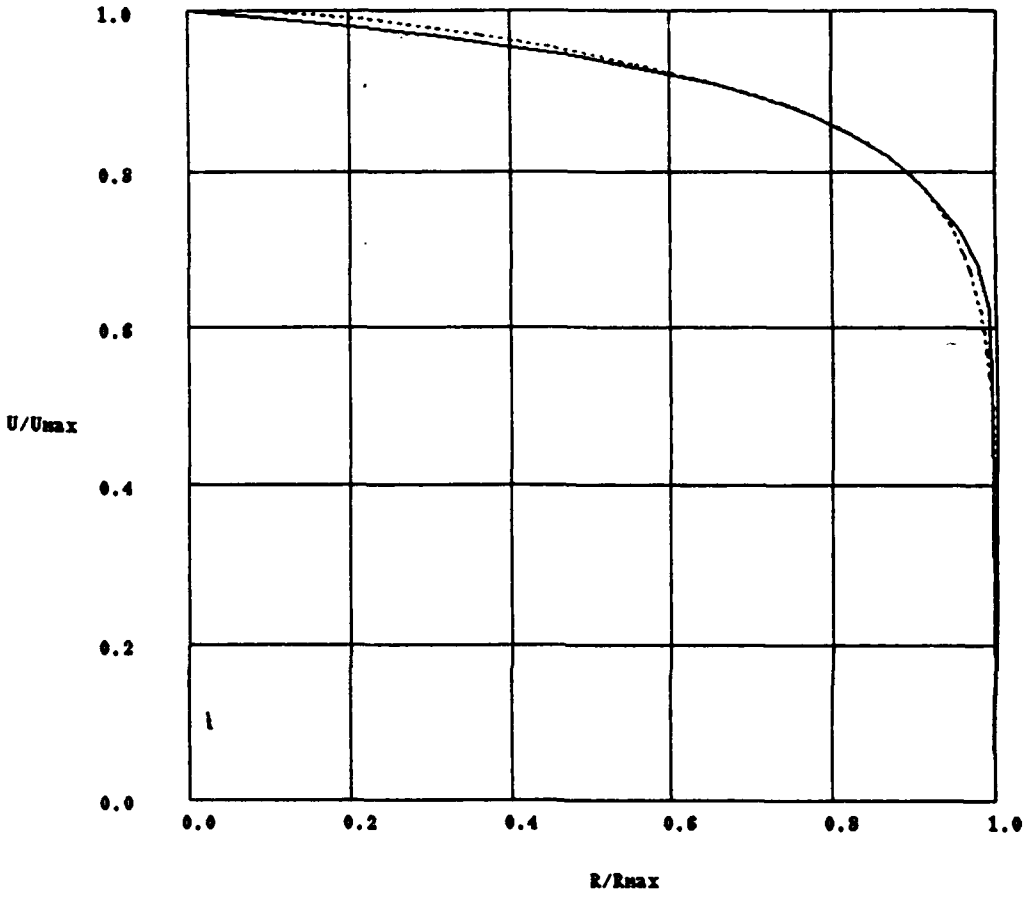


Figure 4. Turbulent pipe flow with Re = 500,000; comparison with Laufer, ——— Laufer, ------ model.

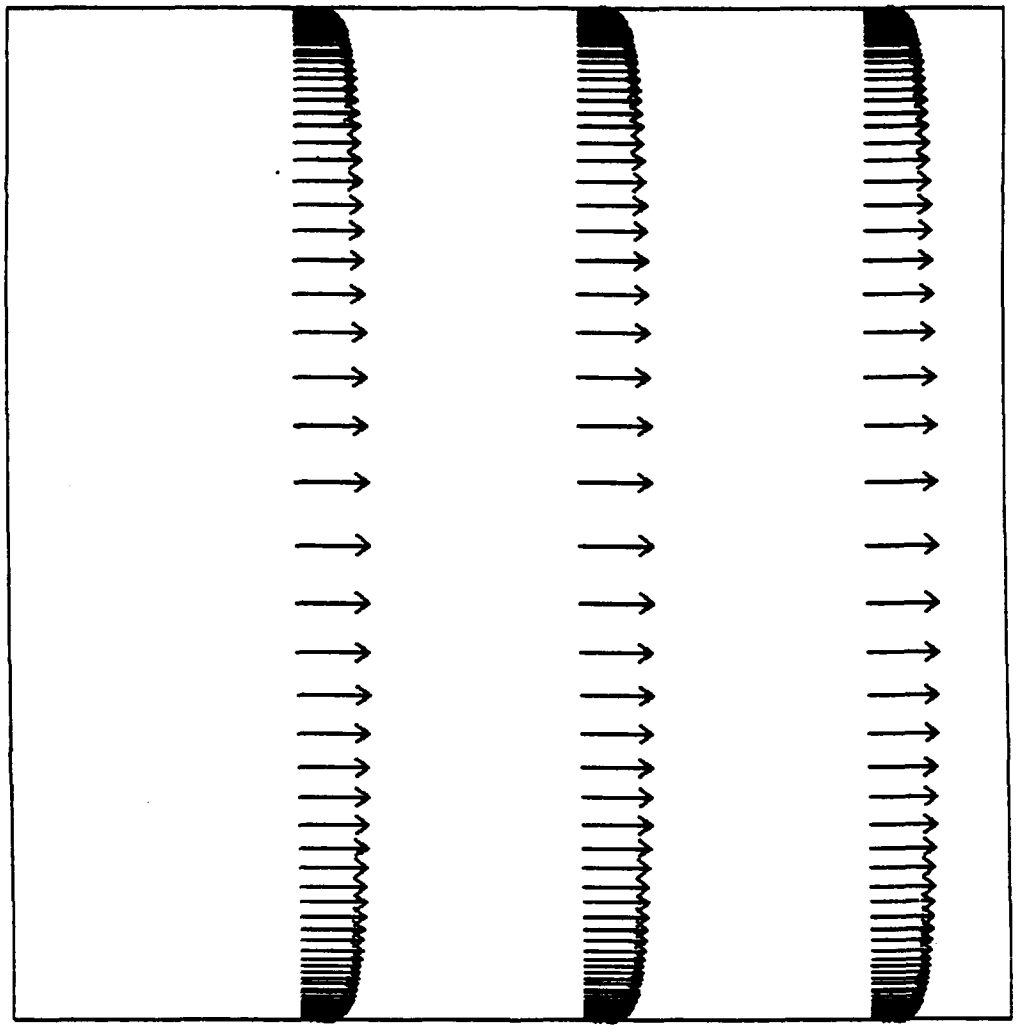


Figure 5. Velocity profile for fully developed flow, Re = 500,000.

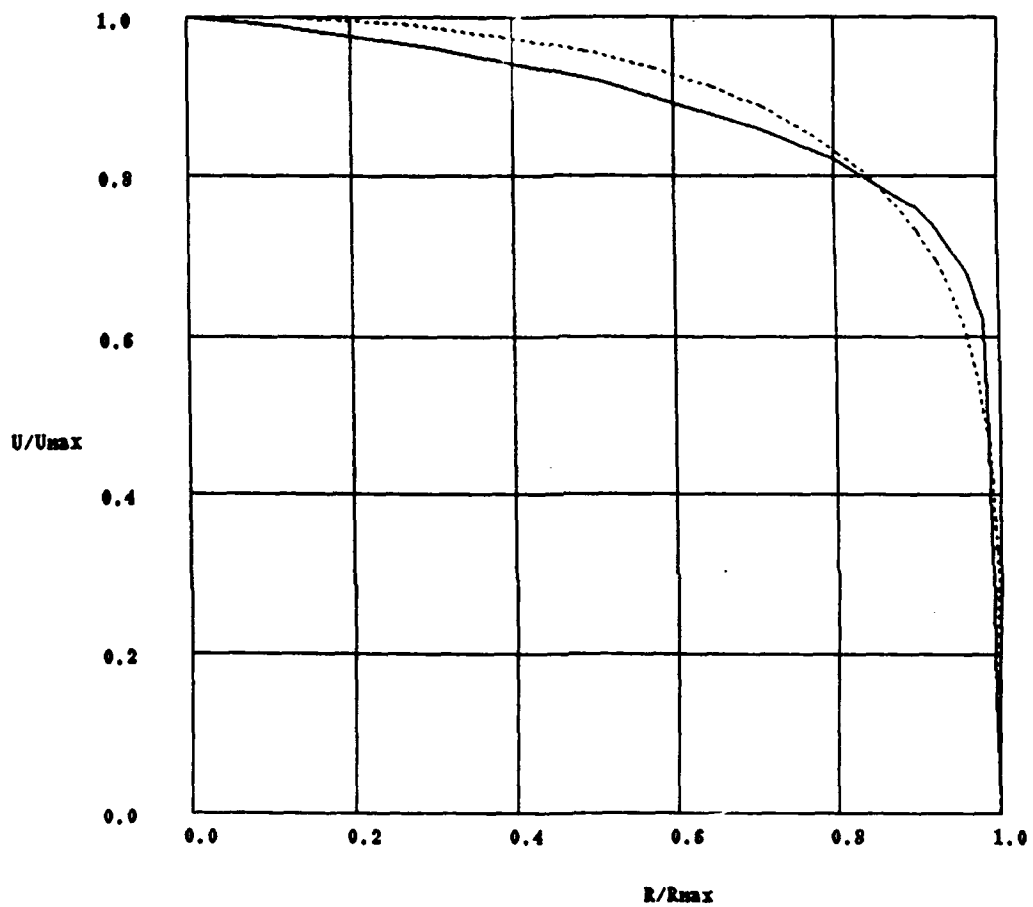


Figure 6. Turbulent pipe flow with Re = 50,000; comparison with power law, ——— power law, ------ model.

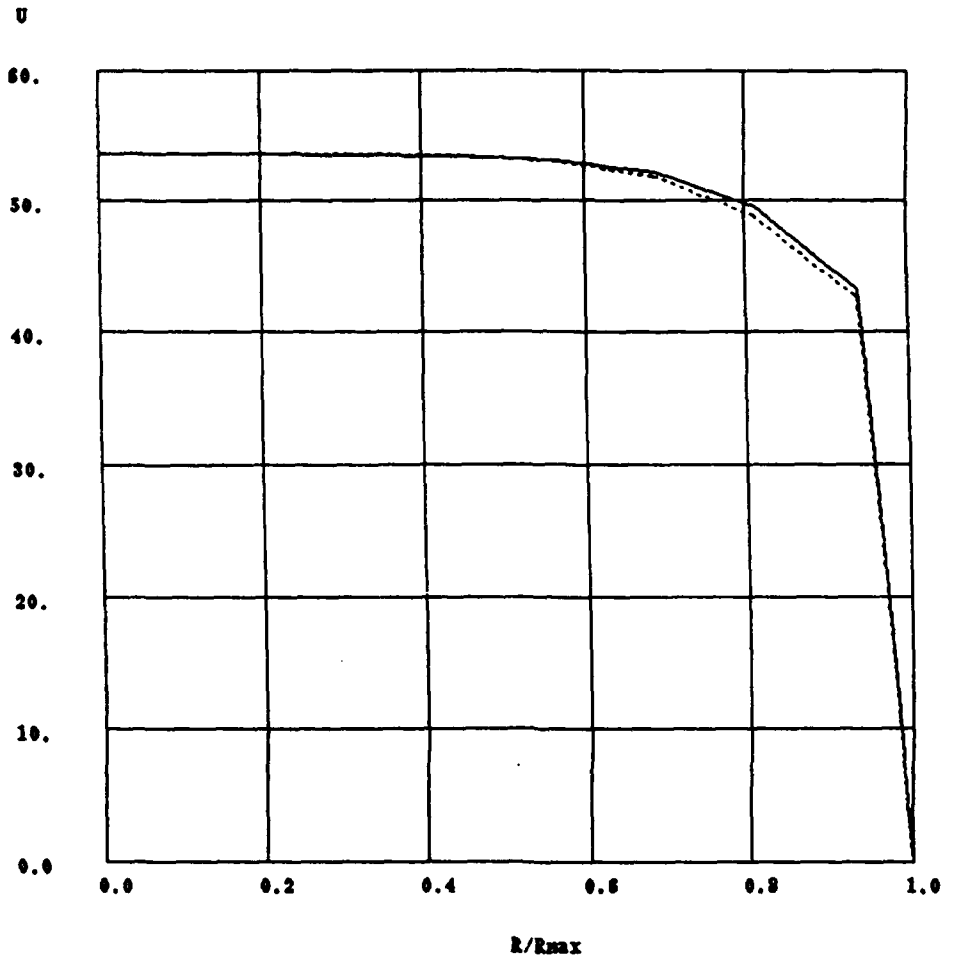


Figure 7. Turbulent pipe flow with Re = 100,100; comparison with Pun and Spalding, ——— Pun, ------ model.

The Pressurized Chamber

The main focus of this project has been the study of the infiltration of contaminant into a pressurized tube with a fabric end cap where the external flow is turbulent. A short tube, pressurized with uncontaminated air, is placed concentrically within a longer, larger diameter tube with contaminated turbulent air flowing through it. The effect of chamber pressure on contaminant infiltration is studied. Because the tubes are circular and located concentrically, the flow is axisymmetric and can be modeled using the two–dimensional equations given earlier. The problem is shown in Fig. 8.

A graded mesh is used to ensure that $Y^* < 30$. To do this a power law formulation is used for the grid spacing. Each direction is divided into subregions as defined by the locations of the wall, as shown in Fig. 9. The subregion's dimension is normalized to a value of 1.0 and and subdivided into n cells. The normalized location of the first coordinate is $s(1) = 0.0$. For subsequent coordinates $s(i+1) = s(i) + 1.0/n$. These evenly spaced coordinates are then mapped onto a graded spacing by $s(i+1) = s(i+1)^p$ for $i = 1,n$. The physical coordinates are found by using the appropriate scale factor and starting value. The subroutines to do this are part of the driver. Several runs are made to find a satisfactory number of cells and grading for each subregion. Fig. 10 illustrates such a mesh in the neighborhood of the chamber. The values finally used are given in Appendix D.

The fabric is very thin compared to the other dimensions in the problem. In order to have a reasonable mesh, it is necessary to use a fictitious thickness for the fabric that is of the order of the cell widths used in the rest of the model. Smearing the fabric's thickness over two cells is accomplished by scaling the physical value by the ratio of actual fabric thickness to the thickness used in the model [13] which is two control volumes. The pressure drop source terms given by eqs.(8) and (9) are adjusted by

$$S^* = SB/2C_v$$

where B is the thickness of the fabric and C_v is the thickness of a control volume or cell.

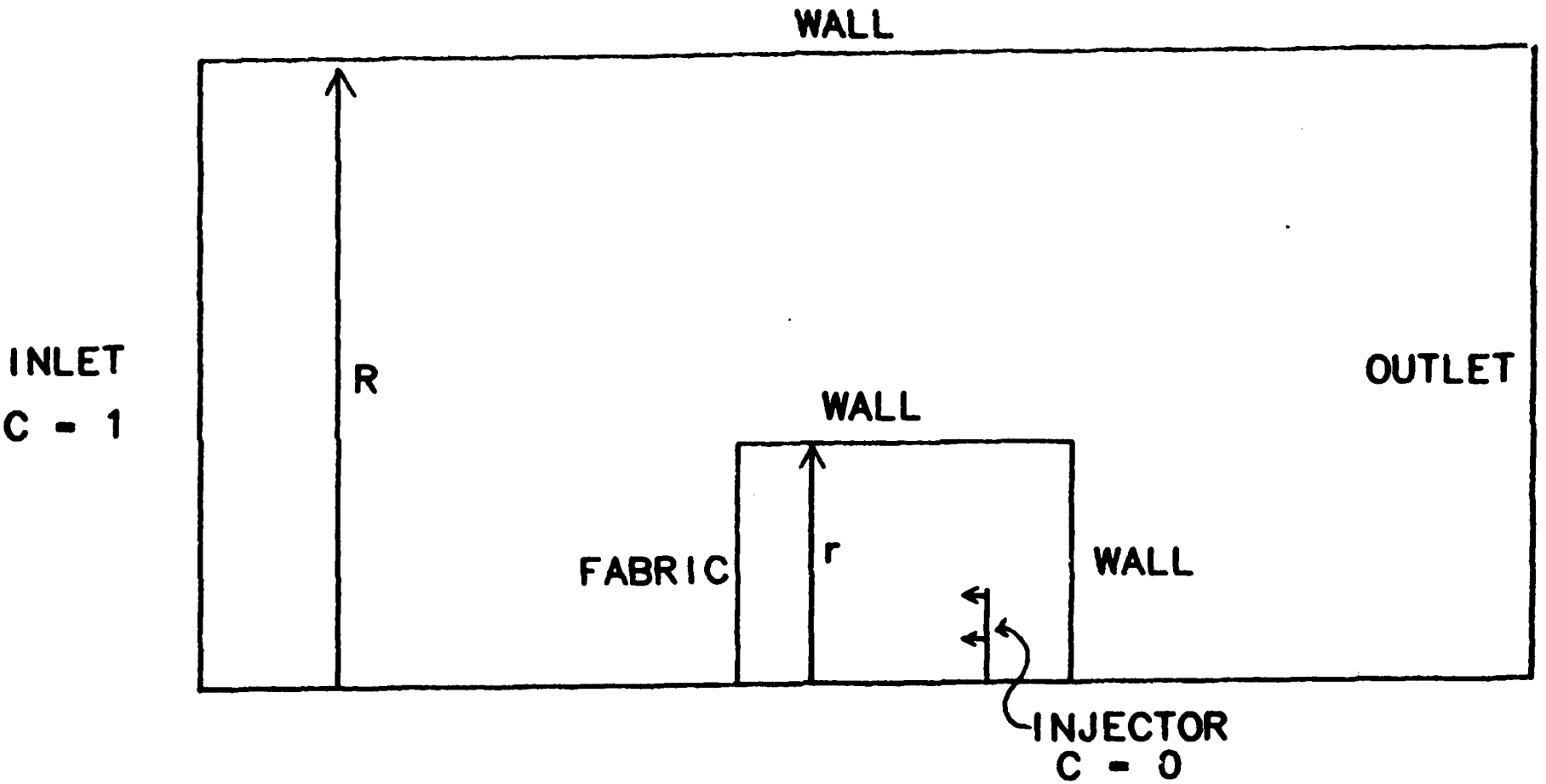


Figure 8. The configuration for the axisymmetric model with R = 4.00 in. and r = 1.5 in.

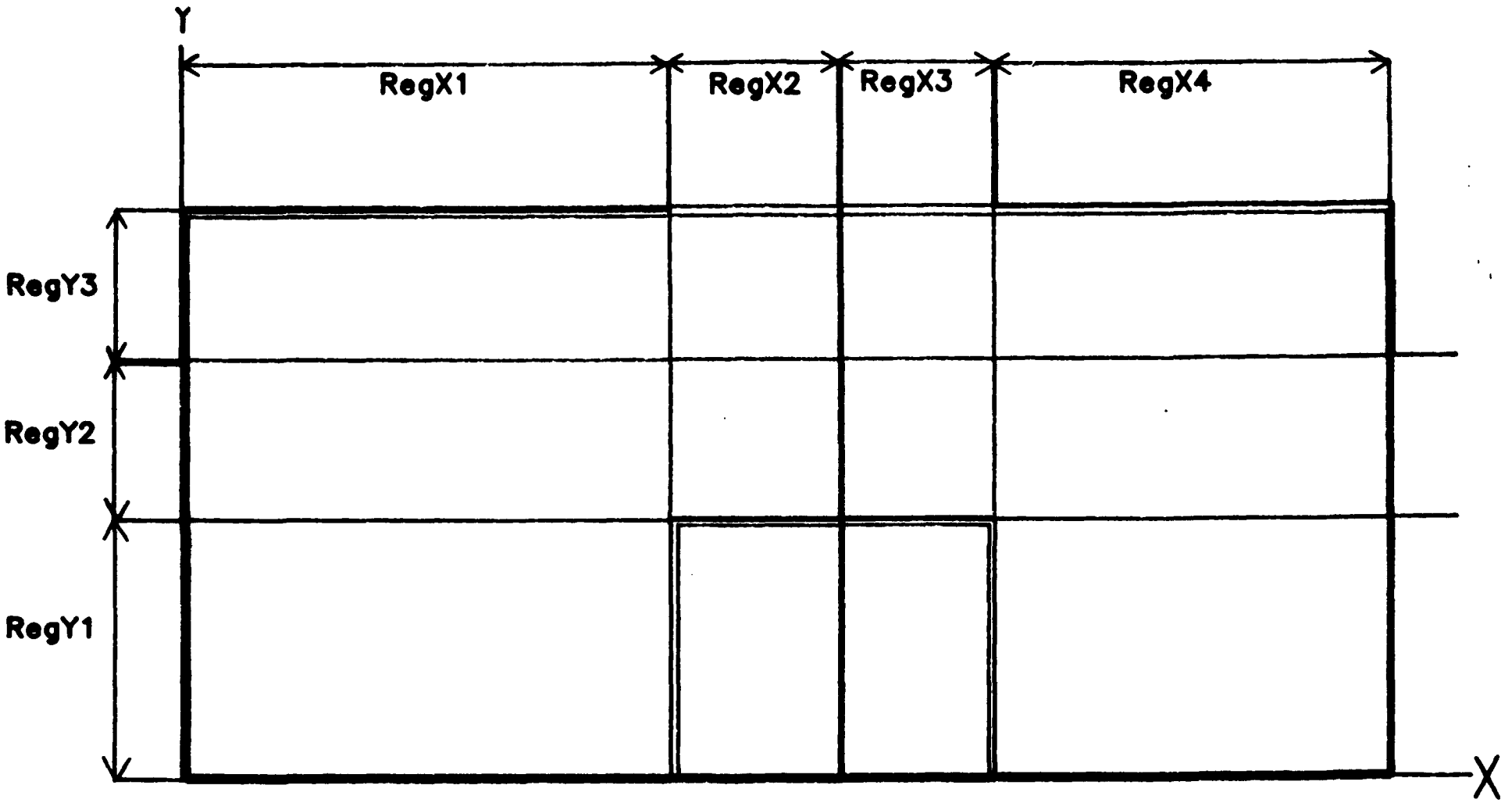


Figure 9. Regions in the x and y directions for mesh subdivision.

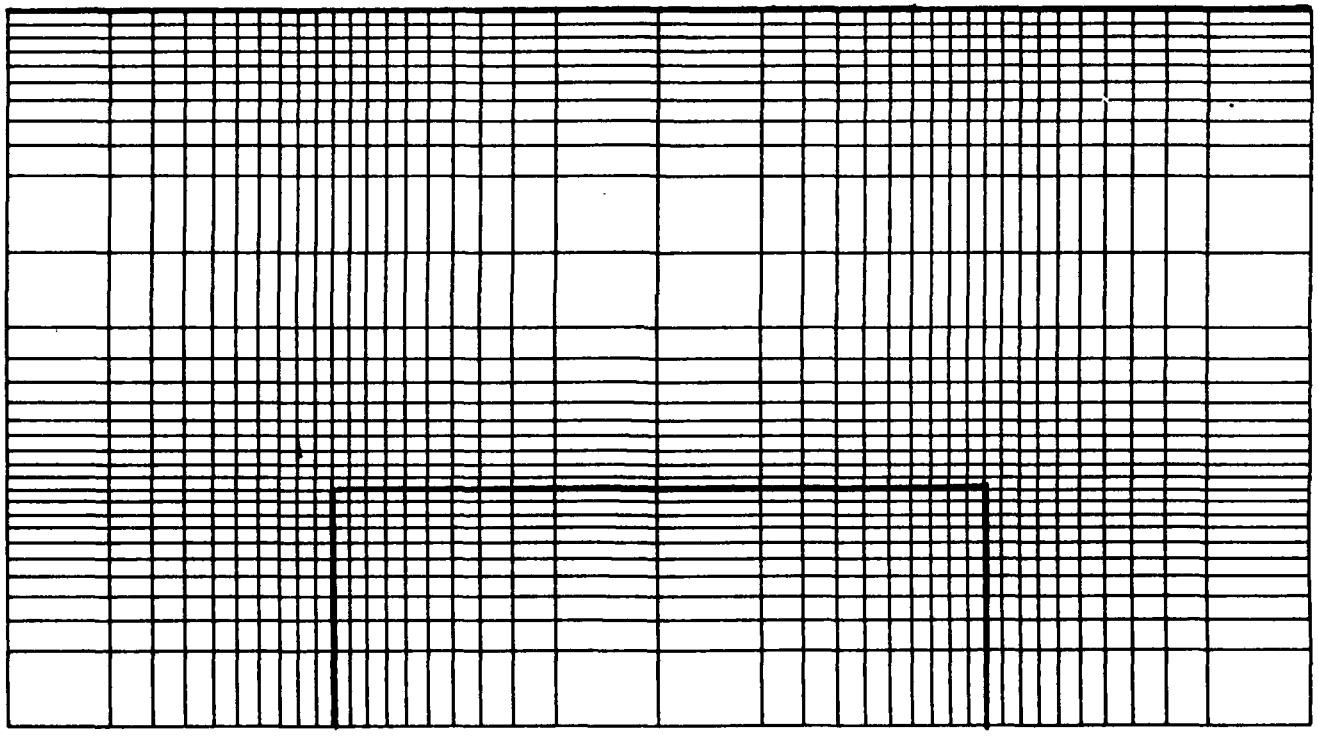


Figure 10. A typical graded mesh.

The diffusion coefficient for the contaminant in eq. (42) is adjusted by

$$\eta^{*} = \eta_{\text{fabric}}(2C_v/B) .$$

Uncontaminated air is injected axially in the negative z–direction over a radial distance of 0.474 in at a distance of 2.0 in from the fabric. Inlet velocities up to 100 in/s were used with the injection velocities ranging from 0.1 to 1.0 times the inlet velocity. The properties for air are $\mu = 0.275\text{e–}8$ lb–s/in^2, $\rho = 0.1063\text{e–}6$ lb–s^2/in^4, and $\eta = 0.044$ in/s^2. The values to be used in eqs.(8) and (9) for the fabric are $\epsilon = 0.43$, A = 149/in, D = 0.00277 in, and the thickness of the fabric B = 0.0268 in. The assumption that the diffusion coefficient for the fiber is negligible yields $\eta_{\text{fabric}} = 0.012$ in/s^2 from eq. (42).

Steady State Analysis

The majority of runs were for steady state analysis since it is the long term performance of the system that is of interest. However, for purposes of validating the model, several transient runs were also made. In all cases, contaminated air enters at the inlet and uncontaminated air is injected into the chamber as shown in Fig. 8. In this section the steady state case is discussed.

Numerous cases were run, a selected few of which will be discussed here. For the first case the inlet velocity is 60 in/s and the injection velocity is 8.0 in/s. Fig. 11 shows the velocity field in the neighborhood of the chamber where the vectors are plotted every other grid point. Fig. 12 shows the flow within the chamber where the vectors are plotted every grid point. Fig. 13 shows the pressure field. Fig. 14 shows the concentration contours for the contaminant. For the second case the injection velocity is increased to 60 in/s. The exterior flow and the pressure field are similar to the previous case. The interior flow is shown in Fig. 15 and the concentration contours are shown in Fig. 16. The last case in this series is for an inlet velocity of 100 in/s and an injection velocity of 60 in/s. Figs. 17 and 18 show the interior flow and the concentration contours, respectively.

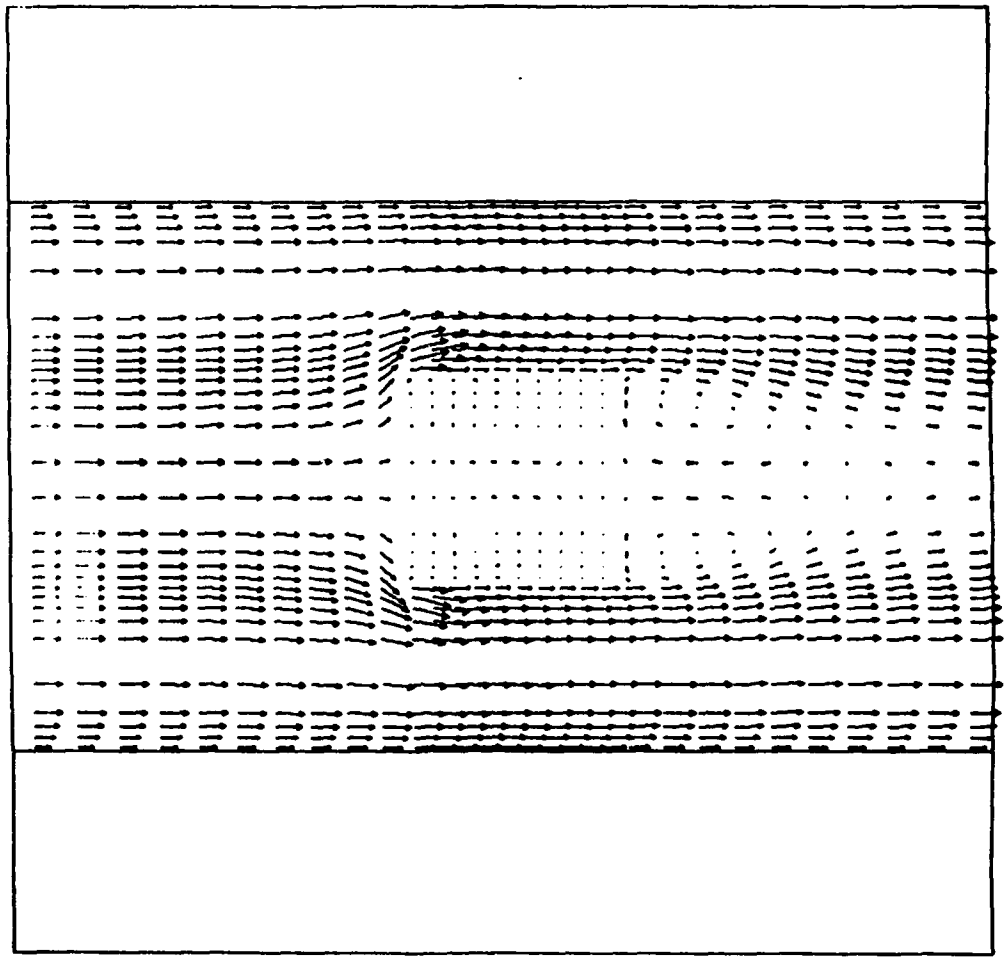


Figure 11. Velocity field for U_{inlet} = 60 in/s and U_{inject} = 8 in/s.

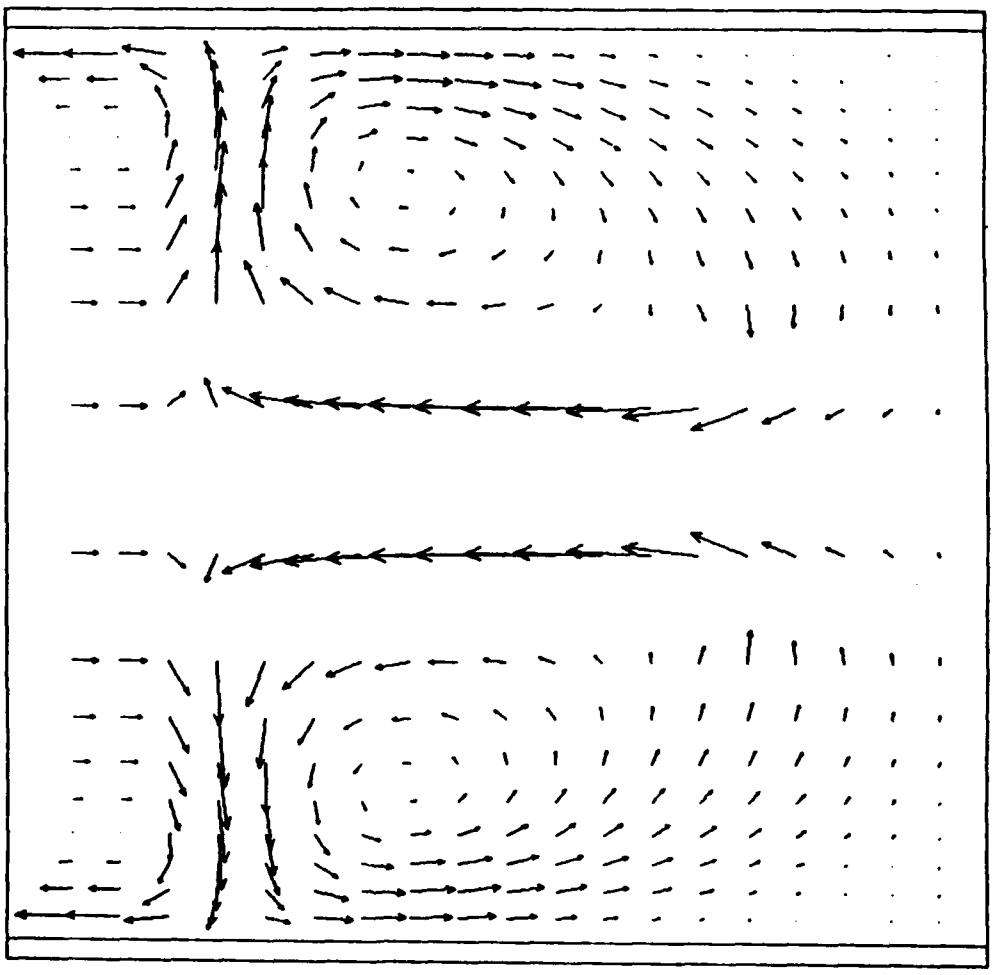


Figure 12. The velocity field within the chamber for the case in Figure 11.

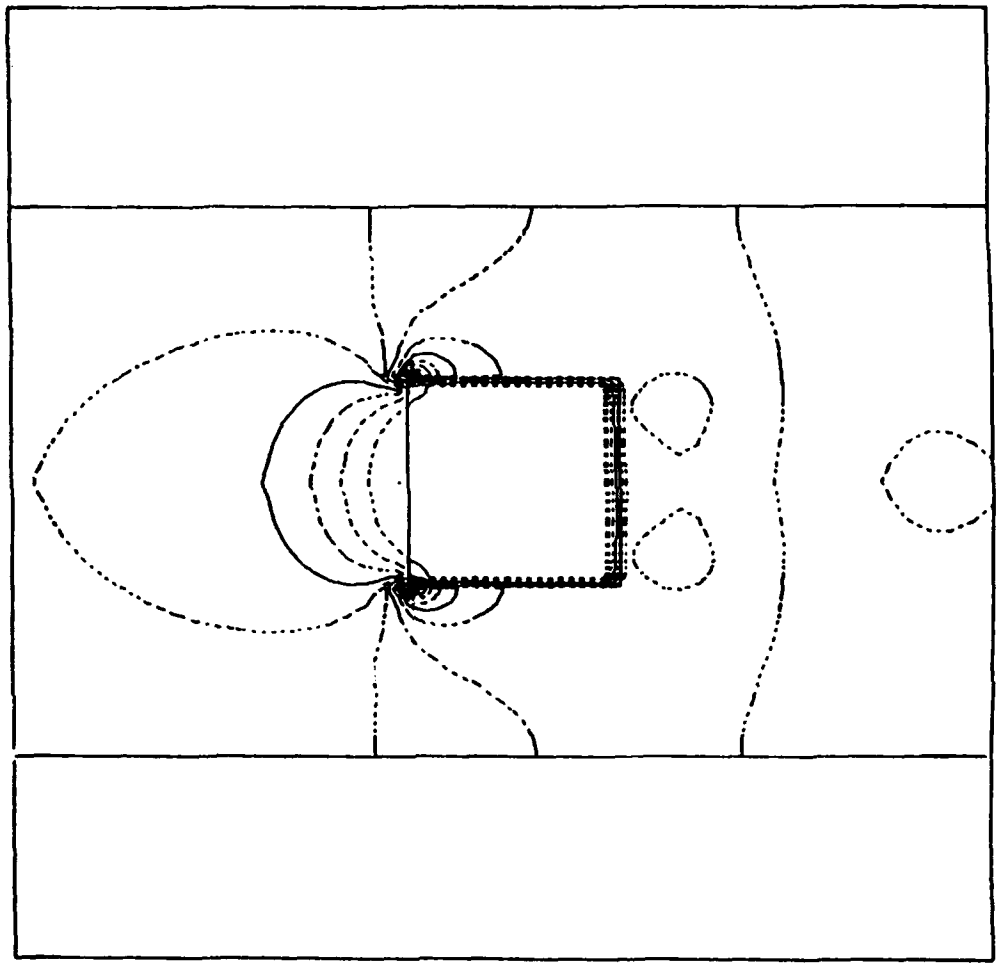


Figure 13. Pressure contours for the case in Figure 11.

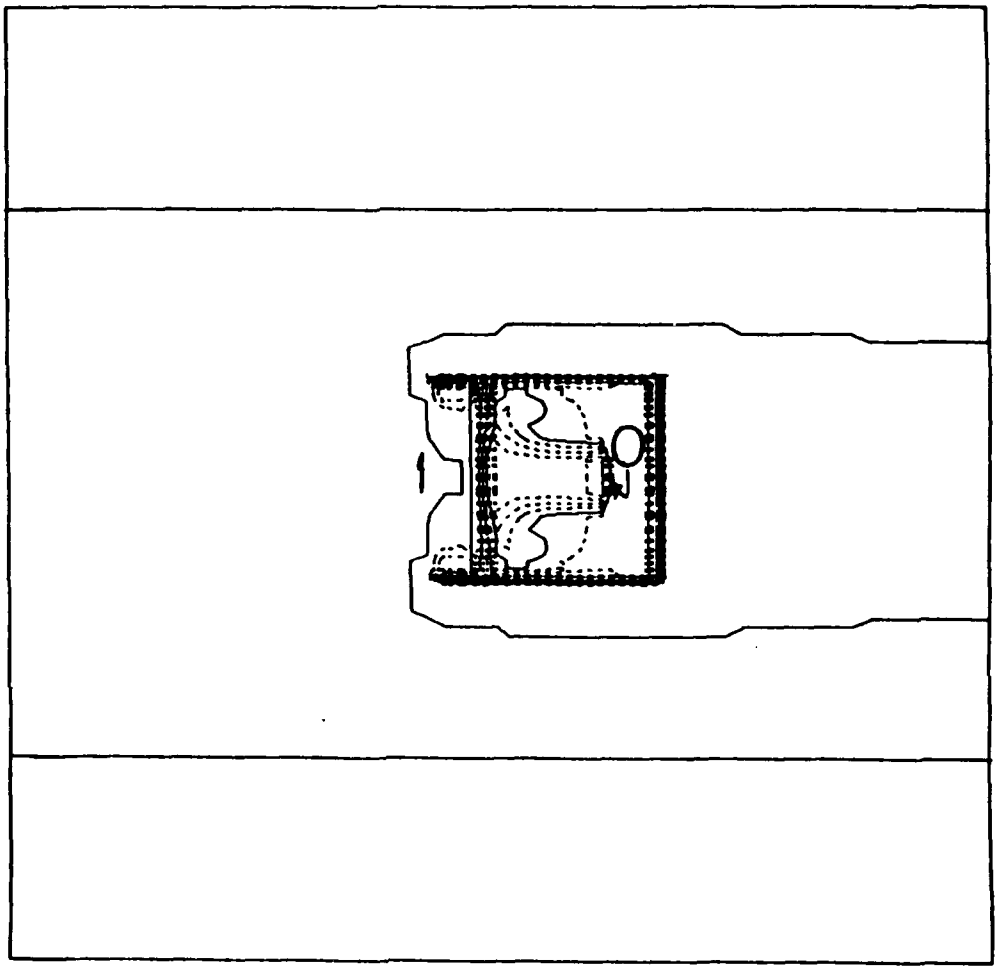


Figure 14. Concentration contours for the case in Figure 11.

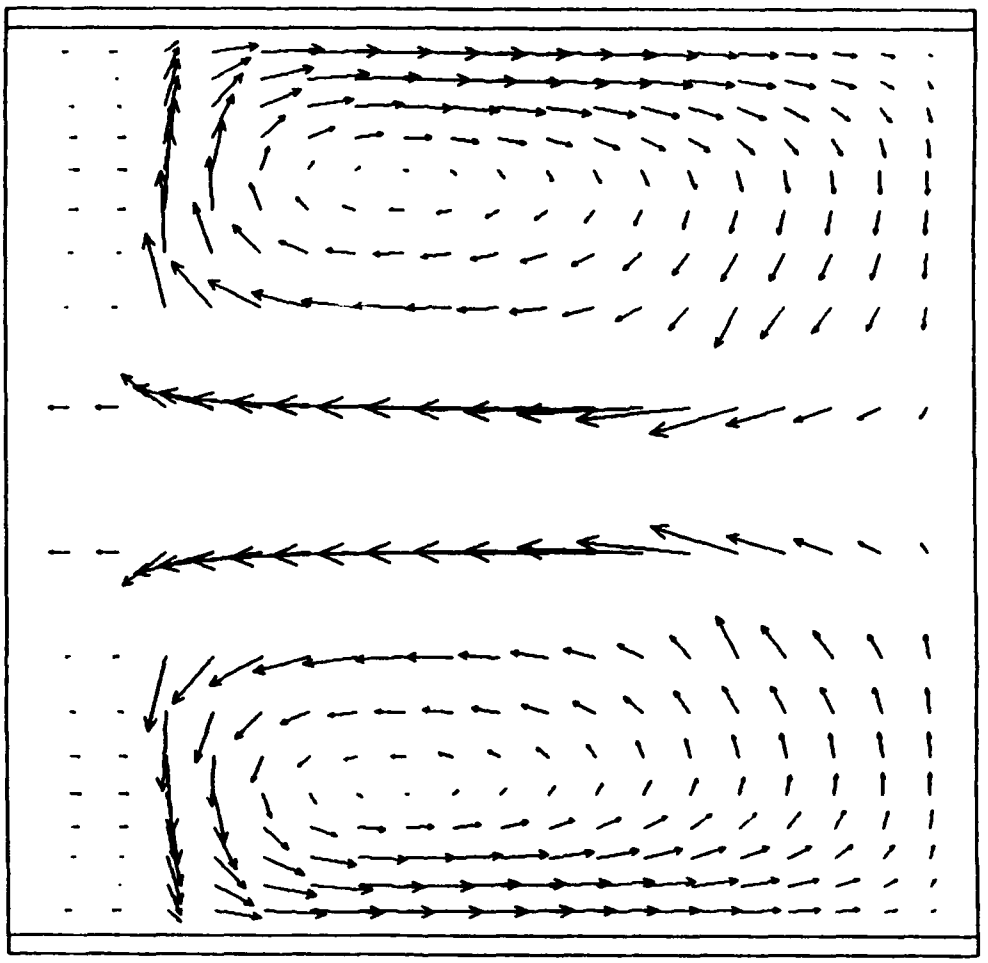


Figure 15. The velocity field within the chamber when U_{inlet} = 60 in/s and U_{inject} = 60 in/s.

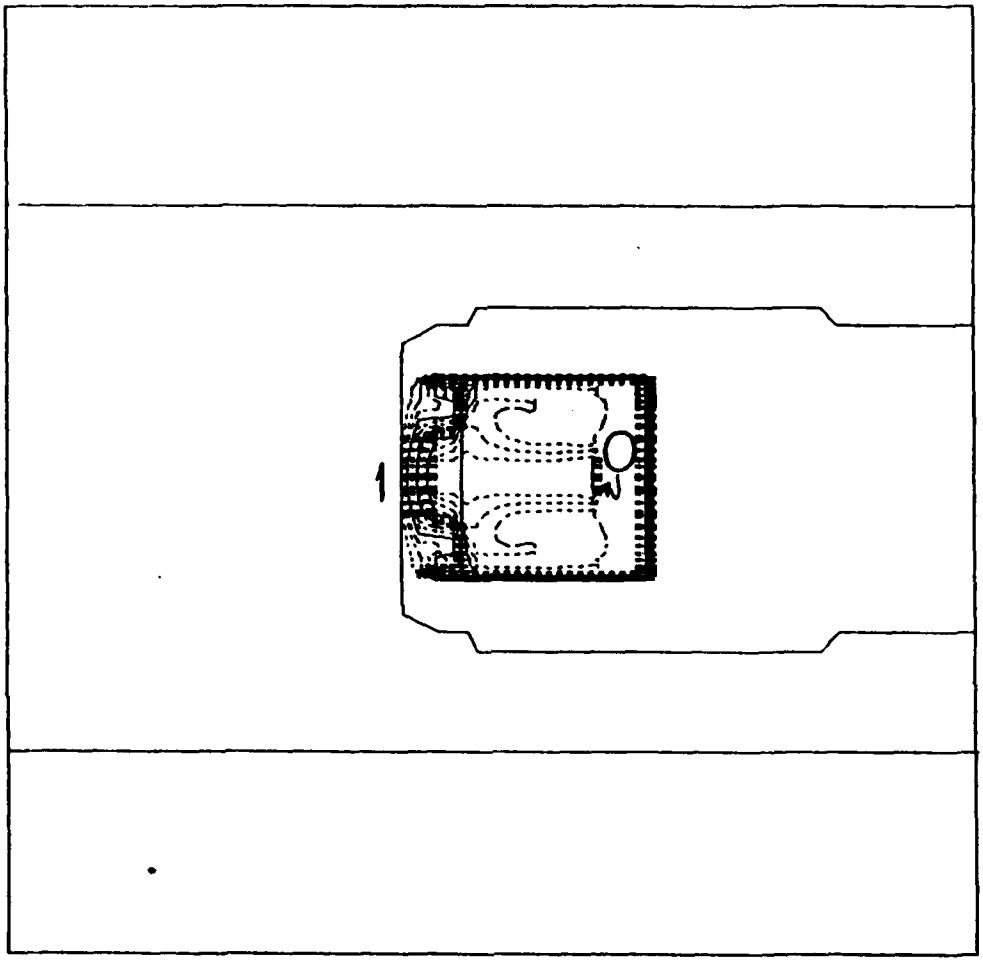


Figure 16. Concentration contours for the case in Figure 15.

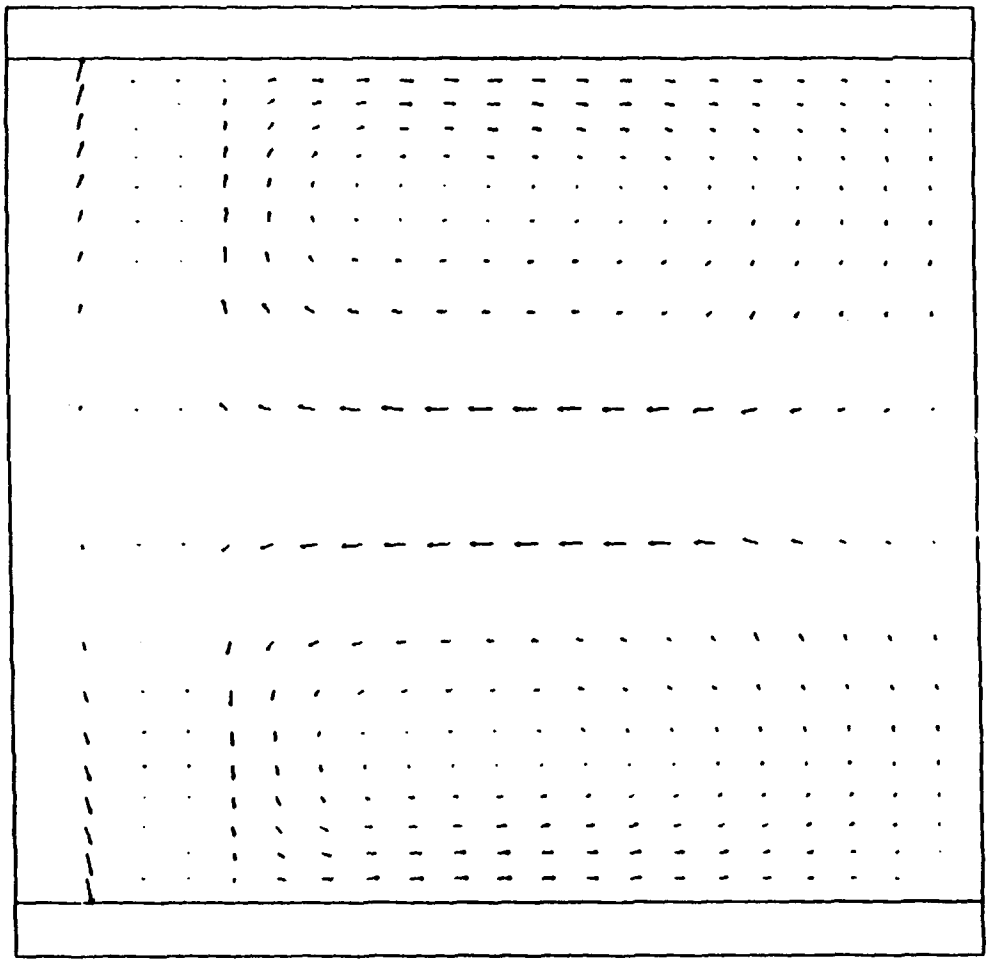


Figure 17. The velocity field within the chamber when U_{inlet} = 100 in/s and U_{inject} = 60 in/s. (the scale is approximately 1/5 that of Fig. 8)

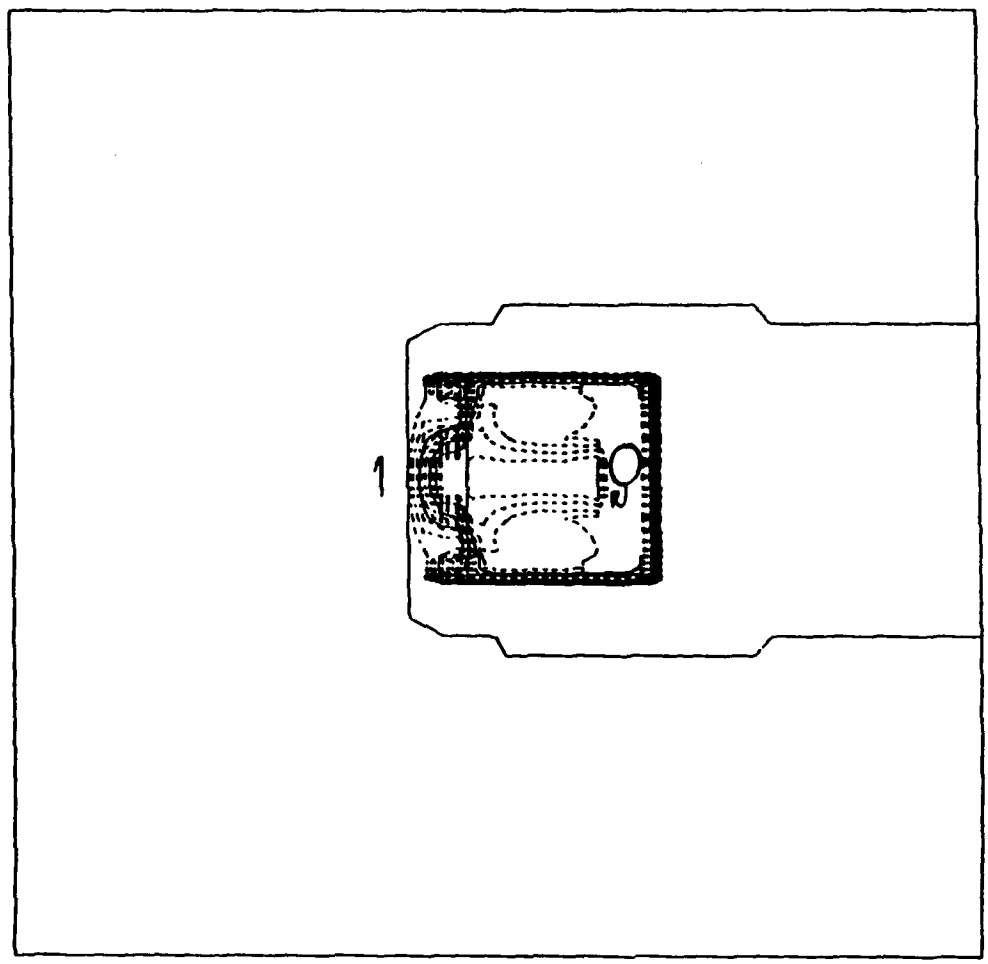


Figure 18. Concentration contours for the case in Figure 17.

Fig. 19 shows the effect of injection velocity on the concentration at three location in the chamber for an inlet velocity of 60 in/s. Initially there is a significant drop in concentration but then it levels off and even increases slightly.

In these four cases it is apparent that the flow field within the chamber is such that it mixes the contaminated air that infiltrates, distributing it over a large portion of the chamber even though there is a net outflow of gas from the chamber. This led to consideration of a different mode of injection of uncontaminated air. Instead of axial injection, radial injection was tried. The injection takes place at a radius of 3.86 in over a length of 1.97 in along the axis starting 0.6 in from the fabric. For an inlet velocity of 50 in/s and an injection velocity of 20 in/s, Fig. 20 shows the flow in the neighborhood of the chamber. Fig. 21 shows the pressure field and Fig. 22 the concentration contours. It is clear that the level of contamination is significantly less for this latter case. The volume flow rate of the injected gas for this case is 2.5 times that for the case of axial injection at 60 in/s so it might be construed that this is the cause of the significantly lower concentration. However, other runs showed that this is not the case.

Transient Analysis

From the point of view of practice, the steady state analysis is a worst case since the concentration of contaminant on the exterior of the fabric will be at its maximum. A transient analysis is important because it illustrates the temporal behavior of the system and can be used as a further check on the validity of the model. In the experimental setup developed in parallel with this study [14], the contaminant is humidity. The velocity is allowed to stabilize before humid air is released into the inlet of the tube. The time variation of the humidity at the inlet and within the chamber is recorded. For the experiment, treated cotton duck was used for the fabric. The characteristics of the fabric needed for the model were measured by Nickerson[15] using micrographs of the test specimens. These are the values previously given. Because the velocity field in the

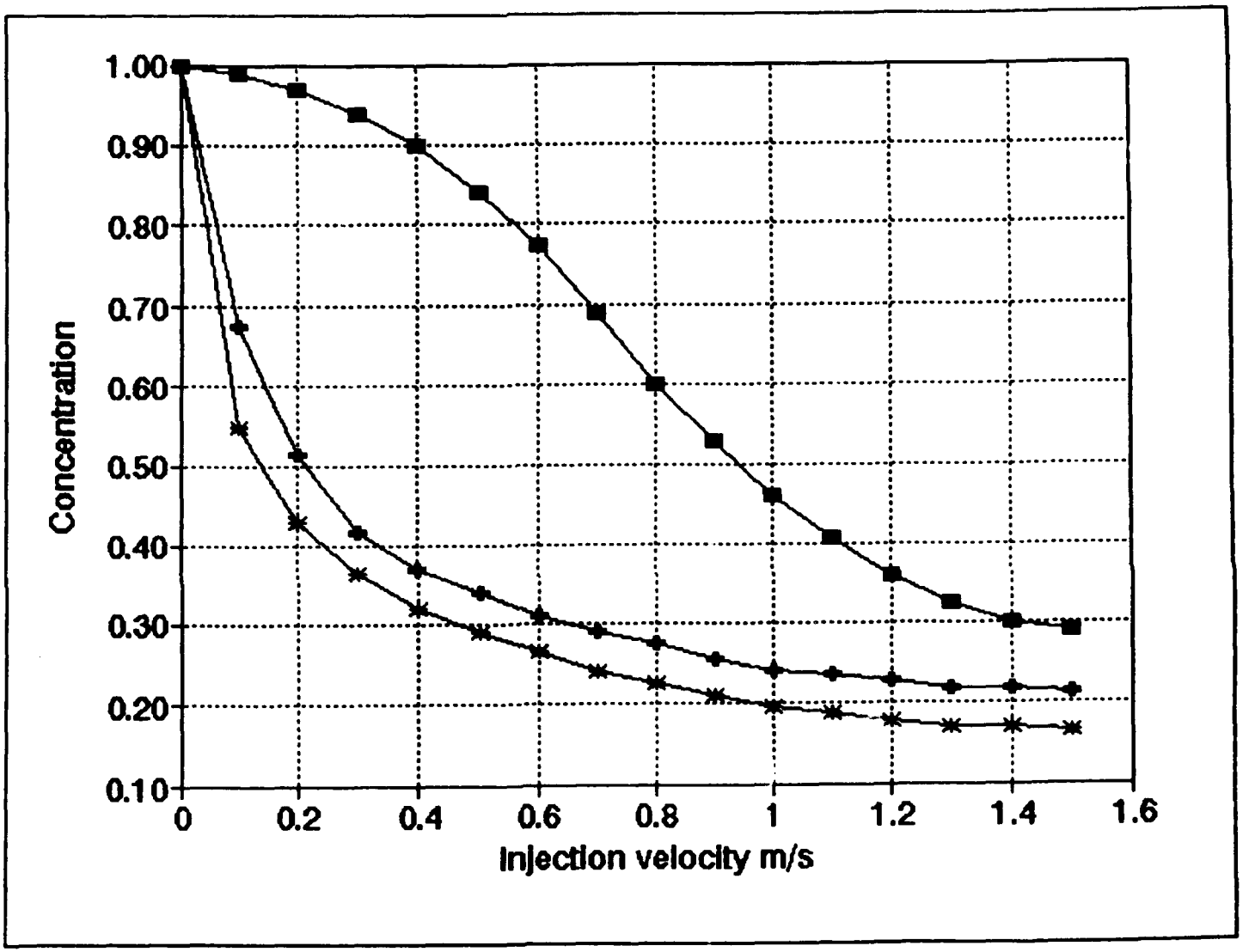


Figure 19. The effect of injection velocity on concentration when U_{inlet} = 60 in/s. The data are for locations 1.18 in from the axis and □ is 0.374 in, * is 1.874 in, and + is 2.776 in from the fabric.

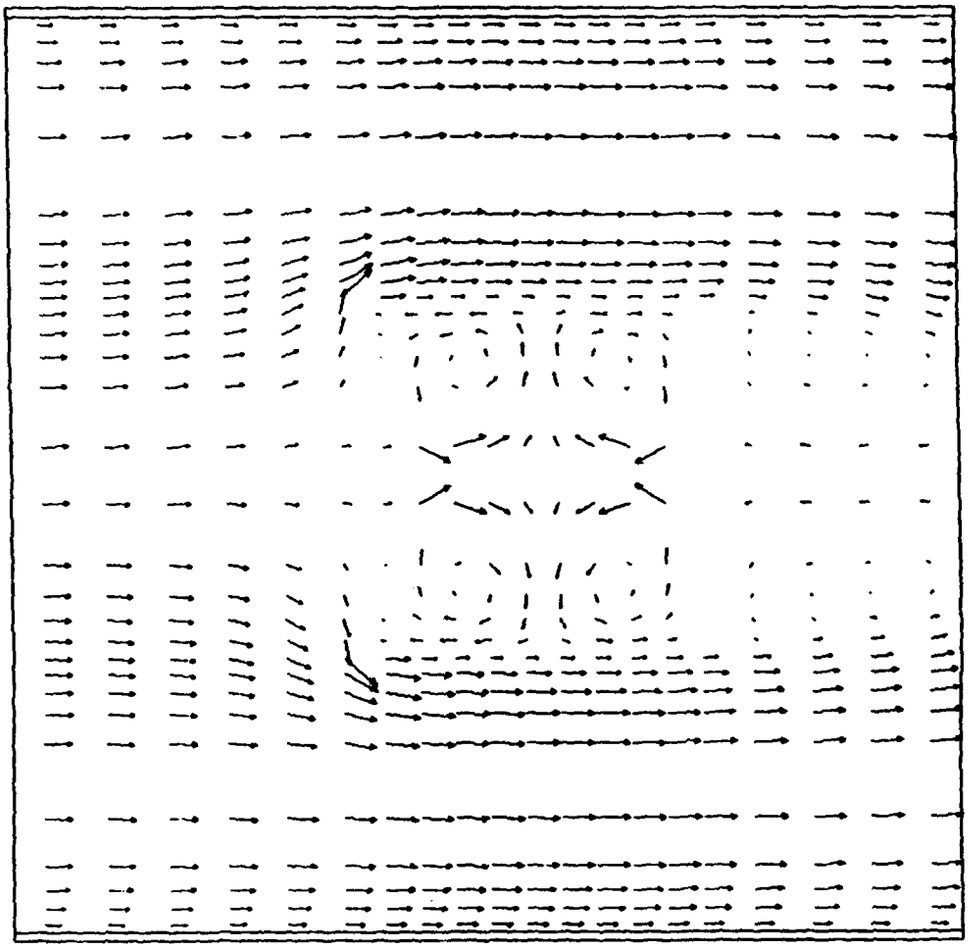


Figure 20. The velocity field for the case of radial injection. U_{inlet} = 50 in/s and V_{inject} = 20 in/s.

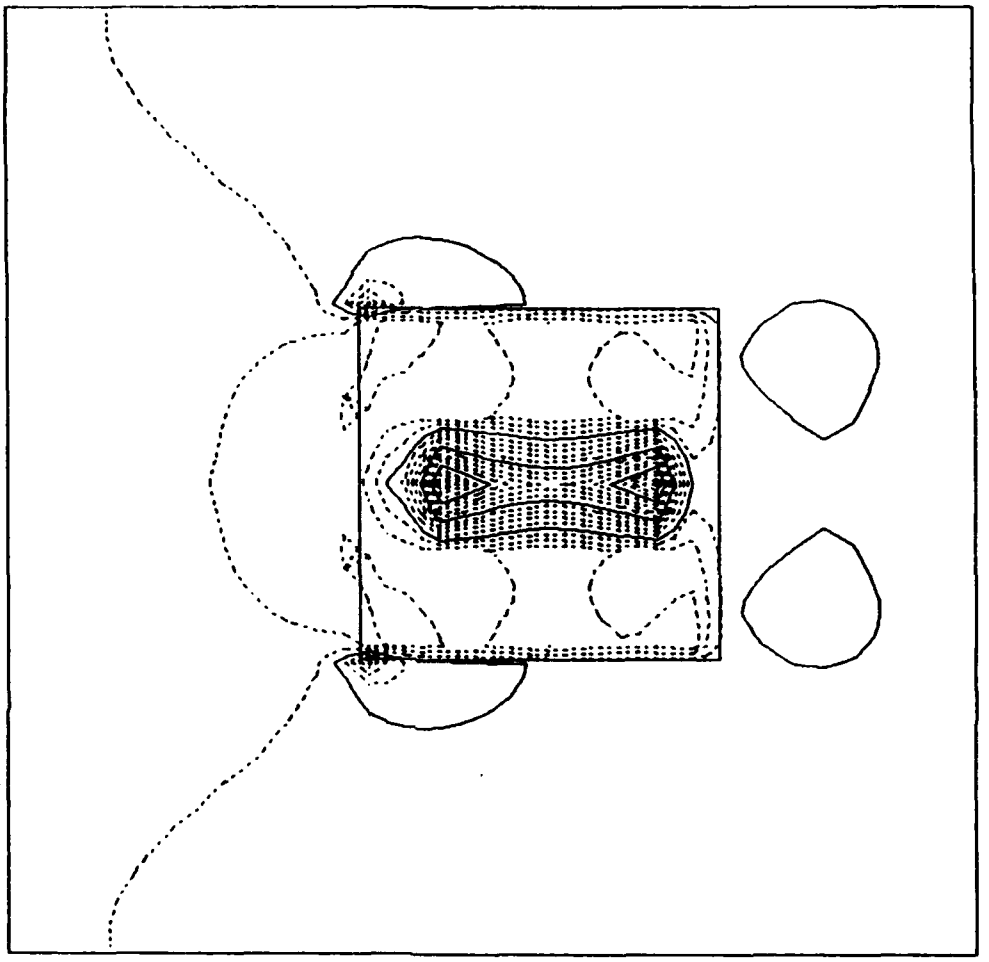


Figure 21. Pressure contours for the case in Figure 20.

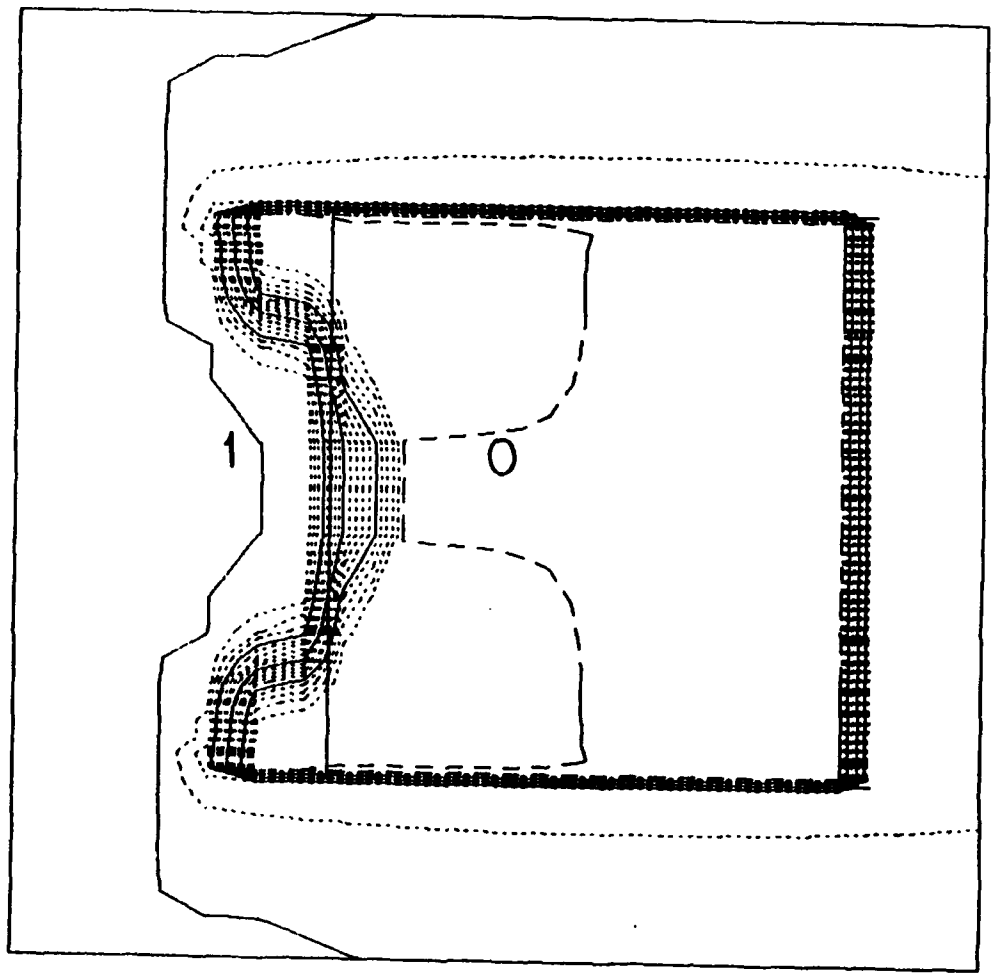


Figure 22. Concentration contours for the case in Figure 20.

experiment was stabilized before the introduction of humid air to the inlet, it can be assumed that only the water vapor concentration is changing in time and that the velocity field is steady. Thus the velocity analyses that were performed for the steady state cases can be used as input for the transient model for the contaminant concentration. Fig. 23 shows the variation with time of inlet concentration used in the model (the variation is $(1.0 - e^{-t/100})$ and is a good representation of the measured data [14]), and the analytically predicted variation within the chamber as predicted by the model for a U_{inlet} = 60 in/s and U_{inject} = 16 in/s. It is clear that the predicted value within the chamber directly tracks the inlet value, as far as can be seen in the graph. This behavior was observed for a wide range of injection velocities and for various locations within the chamber. Fig. 24a shows the experimentally measured response for a fabric covered with the cotton duck for the case of zero injection velocity (no overpressure). Note that the readings are not normalized and that the steady state values are equal, which is to be expected when there is no injection of dry air into the chamber. It should be noted that the measured data are for relative humidity and not for the concentration of water vapor so the comparison can only be qualitative. It is clear that the rise time of the humidity in the chamber is significantly greater than that for the inlet humidity. This large difference in rise times to steady state was troubling. Because the velocity is fairly high, the contaminant is transported quite rapidly from the inlet to the chamber, any change at the inlet being felt within a second or less at the exterior of the chamber. This is very fast compared with the time the solution takes to reach steady–state, which is on the order of 500.0 s. Thus, the model of the fabric appeared to have a serious flaw. In order to check this a simple one–dimensional model of diffusion through a fabric was solved exactly for the case of initial concentration zero in the fabric with one boundary having a concentration equal to 1.0 applied at time zero and the other boundary diffusing through a resistance to a sink at zero concentration. When the resistance is infinite the boundary is

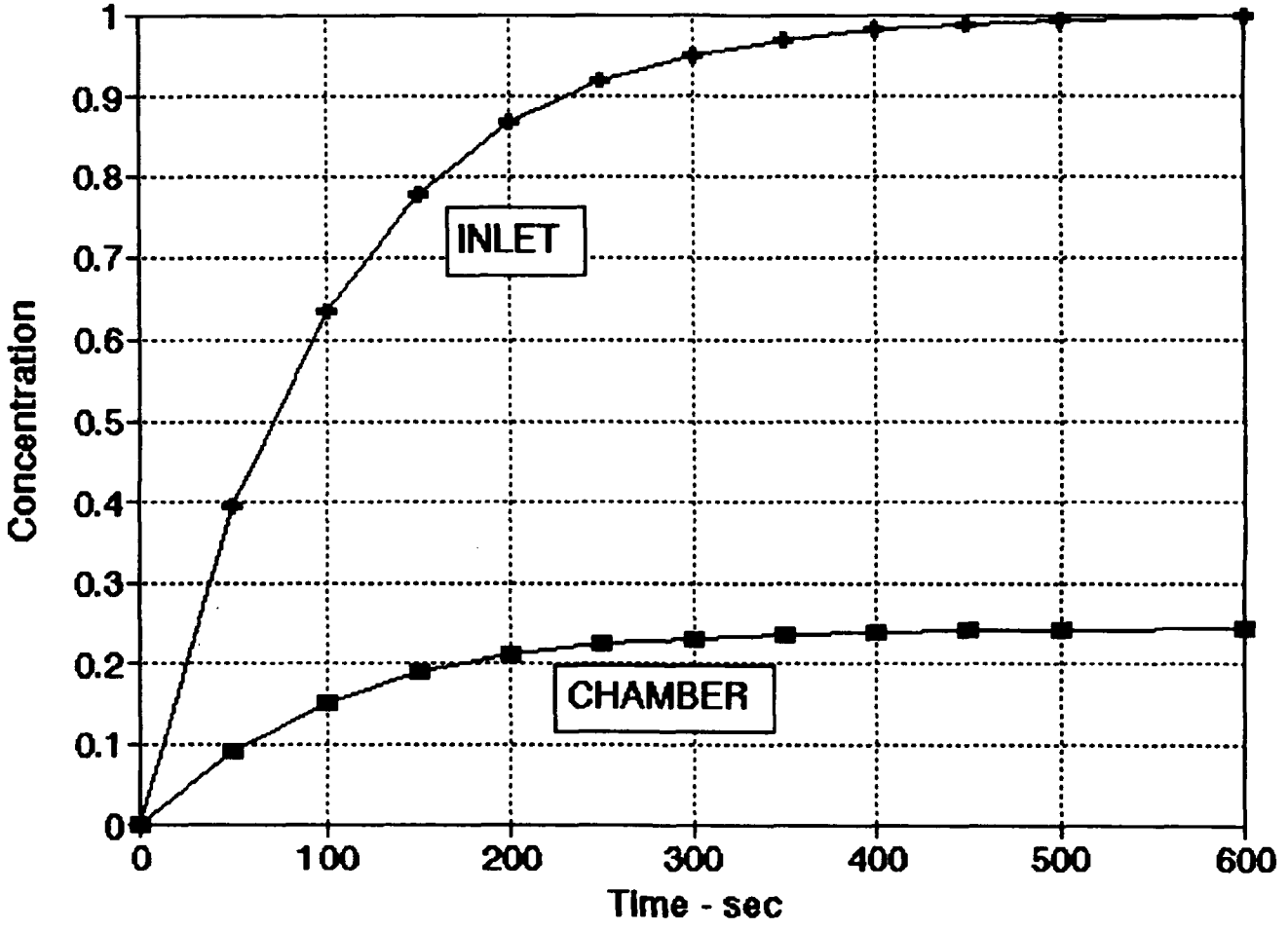


Figure 23. Transient case for U_{inject}= 16 and U_{inlet}= 60 in/s. The location in the chamber is I=64, J=9 (1.875 in from the front of the chamber and 1.298 in from the axis).

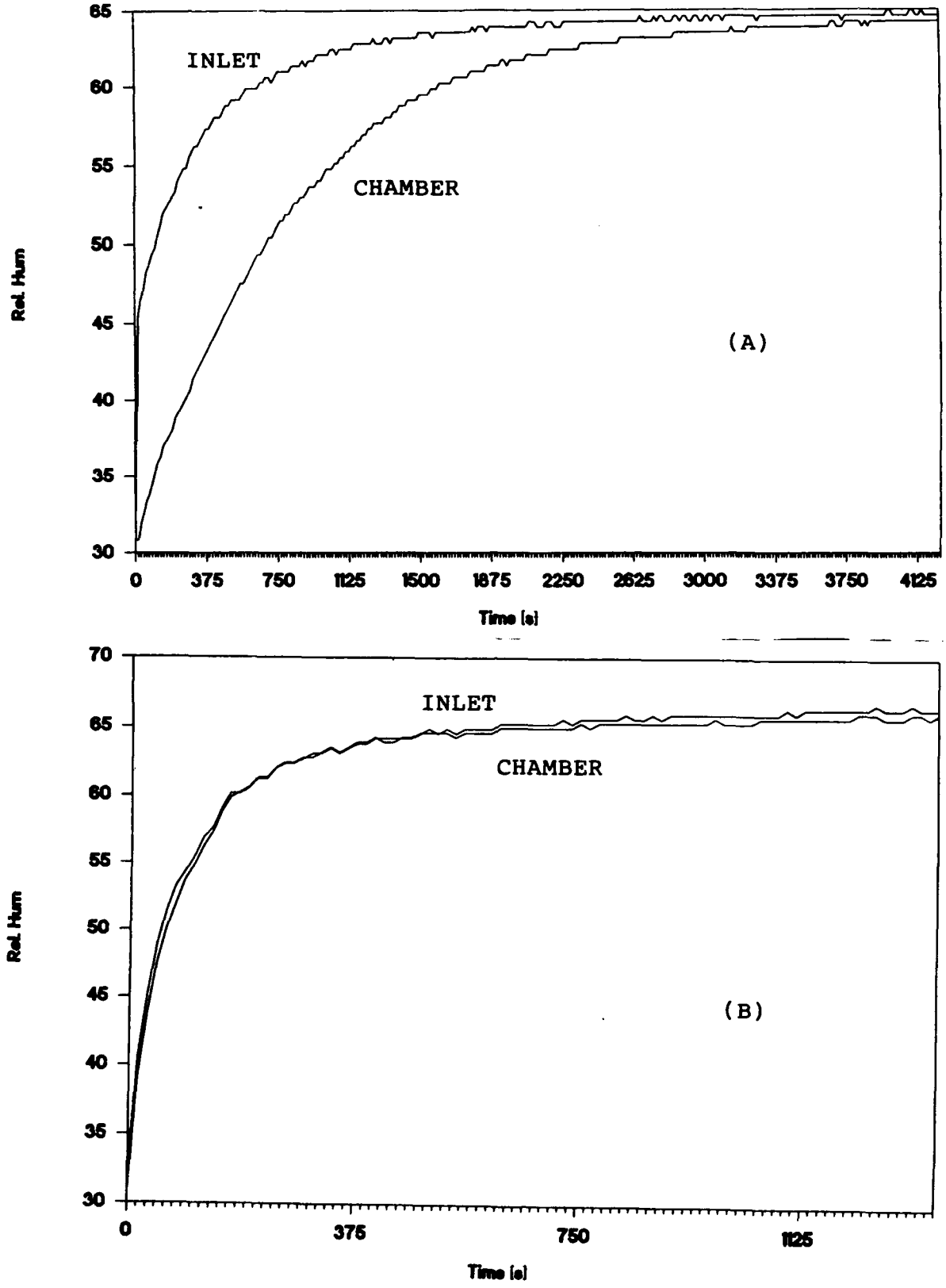


Figure 24. a) Experimental results for cotton duck with no injection into the chamber and $U_{inlet} = 60$ in/s.
b) Experimental results for nylon with no injection into the chamber and $U_{inlet} = 120$ in/s.

impervious and when the resistance is zero the boundary is at zero. The solution for the problem is a series [16] with coefficients that are exponentials in time of the form $e^{-\eta\beta^2 t}$ where β are the roots of $\beta/h + \tan(\beta B) = 0$ and where h is the effective surface conductance at the boundary and B is the thickness of the fabric. The time constant for a typical term in the series is $1/\eta\beta^2$. The time constant gets larger as h gets smaller and the longest time constant in the series corresponds to the first root of the characteristic equation. It can be easily demonstrated that as h gets smaller the first root $\beta \rightarrow \pi/2B$ and for a fabric with $\eta = 0.012$ and $B = 0.027$ that the largest time constant $\tau \rightarrow$ 0.025 s. Steady state can be assumed to be reached in four time constants which is 0.1 s. Thus, the model in the code was, in fact, performing properly under the assumption that the fabric is passive. Further, an earlier study, which compared the fabric model to the work of Armour and Cannon [1,13] showed the model to work properly. The conclusion was that there was another mechanism at work in the experiment, which the model did not account for. It was postulated that the cotton duck was not a passive fabric but one with characteristics that change with increased humidity, i.e., its porosity, surface area–to–volume ratio, pore diameter, and thickness all changing along with the fact that it could be drawing vapor from the air prior to becoming saturated. In order to see if something of this sort was the case a quick test was run with a nylon fabric which was thought to be more stable. Indeed, as shown in Fig. 24b, the test showed that the humidity in the chamber tracked the inlet humidity closely. No values for the fabric parameters for the nylon were available at the time of this study so a model was not attempted.

CONCLUSIONS

The code is a useful tool for modeling turbulent flow in situations with simple geometries, its strength being that it can be used to proof test physical models of such things as fabrics, experimental configurations, the effects of different injection methods, etc. The fact that there were significant differences between the model and experiment appears to be due to the way the fabric was modeled. The model used, however, is the best available. Its weakness is that it assumes the fabric is passive when, in fact, it would appear to be active, i.e., its properties change with conditions. This fact points up the need for further study of fabric behavior in changing conditions to develop models that can be effectively used with codes such as this.

REFERENCES

1. Armour & J. N. Cannon, "Fluid flow through woven screens", AIChE J, v14, 415–419 (1968).
2. A. J. Reynolds, Turbulent Flows in Engineering, J. Wiley & Sons, London (1974).
3. R. B. Bird, W. E. Stewart & E. N. Lightfoot, Transport Phenomena, J. Wiley & Sons, New York (1960).
4. B. E. Launder & D. B. Spalding, "The numerical computation of turbulent flows", Comp. Meth. Appl. Mech. Eng., v3, 269–289 (1974).
5. W. M. Pun & D. B. Spalding, "A general computer program for two–dimensional elliptic flows", Imperial College of Science and Technology, London, U.K., Rept. HTS/76/2 (1977).
6. W. P. Jones & B. E. Launder, "The calculation of low Reynolds number phenomena with a two–equation model of turbulence", Int. J. Heat & Mass Trans., v16, 1119–1130 (1973).

7. B. E. Launder & D. B. Spalding, Mathematical Models of Turbulence, Academic Press, London (1972).

8. S. V. Patankar & D. B. Spalding, Heat and Mass Transfer in Boundary Layers, Intertext Books, London (1970).

9. B. D. Agarwal & L. J. Broutman, Analysis and Performance of Fiber Composites, J. Wiley & Sons, New York (1980).

10. S. V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere, Washington D. C. (1980).

11. S. R. Robertson, "The Implementation of the SIMPLER algorithm for the Dispersal of Agent Within Enclosures", U. S. Army Natick R.D. & E. Center Contract Rept. No. DAAG29–81–D–0100 (1986).

12. J. Laufer, "The structure of turbulence in fully developed pipe flow", NACA Rept. 1247 (1953).

13. S. R. Robertson, "Contaminant infiltration into a pressurized structure with fabric walls", Appl. Math. Modelling, v13, 369–373 (1989).

14. G. Vincens & K. Welch, private communication, ETD–AMED Natick R.D. & E. Center (1991).

15. C. Nickerson, private communication, ETD–AMED Natick R.D. & E. Center (1990).

16. H. S. Carslaw & J. C. Jaeger, Conduction of Heat in Solids, Oxford University Press, London (1959)

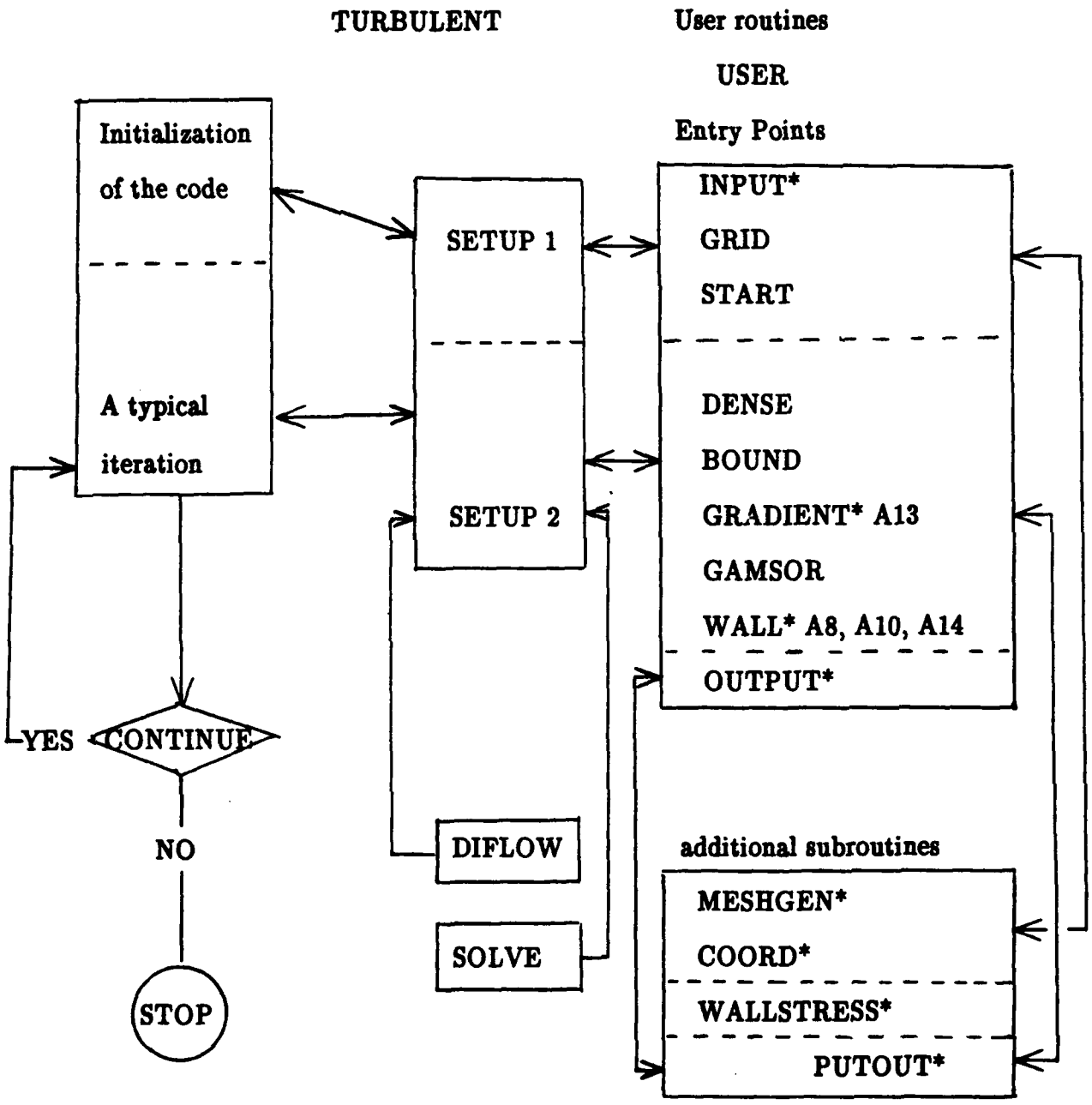
This document reports research undertaken at the US Army Natick Research, Development and Engineering Center and has been assigned No. NATICK/TR–93/009 in the series of reports approved for publication.

APPENDIX A
Simpler Code

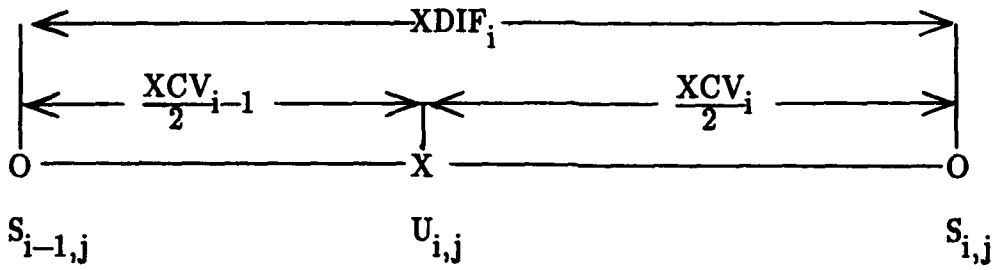
This code is a modification of an earlier SIMPLER code for laminar flow. It is called, simply enough, TURBULENT. It has been modified to permit the solution of turbulent flow problems for incompressible fluids. A flow chart is given on the next page with the important differences starred. The pages of the code on which the changes occur are indicated. The pages for the main code (the solution engine) are numbered 61–74. The changes are indicated by an arrow. The changes are merely calls to ENTRY points in the USER routine (DRIVER) that perform computations peculiar to the two equation model of turbulent flow. In addition, several other subroutines have been developed. The parts that the analyst is responsible for appear on the right side of the following flow chart. Not all the user–developed subroutines indicated in the chart are used for every problem studied.

Three input–output devices are opened in the main program and one in the user routine. Device 15 uses a file called "INPUT" for input, device 16 uses a file called "OUTP" for printed output, device 17 uses a file called "PLOT15" for formatted output to be used with the contour and vector plotting programs, and device 12 uses a file called "RESTART" both for restarting long runs and for saving information from a velocity analysis for use in a mass transport analysis of contaminant.

A flow chart showing the relation of the user routines to the main program TURBULENT.



A word of explanation about interpolation from main grid points to velocity grid points is required. In particular, on page 18 the shear stress is interpolated from the main grid points to the u–velocity grid points. Refer to the following figure,



Where S is a scalar given at the main grid points O and is interpolated to the velocity grid point X by

$$S_{i,j}^{(X)} = S_{i-1,j} + XCV_{i-1}(S_{i,j} - S_{i-1,j})/2\cdot XDIF_i$$
$$= (XCV_{i-1}/2\cdot XDIF_i)\cdot S_{i,j} + (1 - XCV_{i-1}/2\cdot XDOF_i)\cdot S_{i-1,j}$$
$$= FX_i\cdot S_{i,j} + FXM_i\cdot S_{i-1,j} \ .$$

Similarly for the y–direction.

FORTRAN VARIABLES IN SIMPLER ALGORITHM

AIM(I,J)	Finite difference coefficient A_W
AIP(I,J)	Finite difference coefficient A_E
AJM(I,J)	Finite difference coefficient A_S
AJP(I,J)	Finite difference coefficient A_N
APT	Unsteady flow term $\rho / \Delta t$
ARX(J)	Area of the main c.v. face normal to x direction
ARXJ(J)	The part of ARX(J) that overlaps the c.v. for V(I,J)
ARXJP(J)	The part of ARX(J) that overlaps the c.v. for V(I,J+1)
CON(I,J)	The constant term in the f.d. equation, also used for S_c in GAMSOR in USER routine
DIFF	Diffusion term GAM*area/distance input to DIFLOW
DT	Time step for transient problems
DU(I,J)	D^u influencing U(I,J)
DV(I,J)	D^v influencing V(I,J)
F(I,J,NF)	Array containing dependent and auxiliary variables
FLOW	Mass flow rate through a c.v. face
FV(J) and FVP(J)	permit interpolation from the V grid to the main grid through relations of the form

$$A(I,J)_{main} = FV(J)*A(I,J)_{v\ grid} + FVP(J)*A(I,J+1)_{v\ grid}$$

FX(I) and FXM(I)	as above but interpolates from main to U grid
FY(I) and FYM(I)	as above but interpolates from main to V grid
GAM(I,J)	The diffusion coefficient
I	Index denoting position in x
IPREF	The I value of the reference pressure point
IST	The first internal point value of I
ISTF	IST - 1
ITER	Iteration counter

J	Index denoting position in y
JPREF	The J value of the reference pressure point
JST	The first internal point value of J
JSTF	JST - 1
LAST	Maximum number of iterations
LPRINT(NF)	when .TRUE. , F(I,J,NF) if printed
LSOLVE(NF)	when .TRUE. a P.D.E. is solved for NF
LSTOP	When .TRUE. computation stops
L1	Number of grids in x direction
L2	L1 - 1
L3	L1 - 2
MODE	Co-ordinates 1=planar, 2=cylindrical, 3=spherical
M1	Number of grids in y direction
M2	M1 - 1
M3	M1 - 2
NF	Index denoting the variable
NFMAX	Largest value of NF for which storage is assigned
NGAM	NFMAX+2; GAM(I,J) is equivalent to F(I,J,NGAM)
NRHO	NFMAX+1; RHO(I,J) is equivalent to F(I,J,NRHO)
NTIMES(NF)	Number of sweeps in SOLVE at each iteration for each dependent variable
P(I,J)	The pressure, p
PC(I,J)	The pressure correction, p'
PT( ), QT( )	Matrices used in TDMA
R(J)	The radius r for the main grid point I,J
RELAX(NF)	Relaxation parameters
RHO(I,J)	The density

RHOCON	The density for constant density problems
RMN(J)	The value of r at V(I,J)
SMAX	The largest absolute value of the mass error in a c.v.
SSUM	Algebraic sum of all the mass errors
SX(J)	Scale factor for x direction at Y(J)
SXMN(J)	Scale factor for x direction at YV(J)
TIME	Time for unsteady problems
TITLE(NF)	Title of variable for output purposes
U(I,J)	The x direction velocity
V(I,J)	The y direction velocity
VOL	Volume of the c.v.
X(I)	Values of the x direction grids
XCV(I)	Size of main c.v. in the x direction
XCVI(I)	Part of XCV(I) that overlaps c.v. for U(I,J)
XCVIP(I)	Part of XCV(I) that overlaps c.v. for U(I,J+1)
XCVS(I)	As XCV(I) but for U grid
XDIF(I)	X(I) - X(I-1)
XL	x direction length of integration region
XU(I)	x locations of U(I,J)
Y(J)	Values of the y direction grids
YCV(J)	Size of main c.v. in the y direction
YCVR(J)	Same as ARX(J)
YCVRS(J)	Area of v c.v. face normal to x direction
YCVS(J)	As YCV(J) but for V grid
YDIF(J)	Y(J) - Y(J-1)
YL	y direction length of integration region
YV(J)	y locations of V(I,J)

```
C
C.....1.....2.....3.....4.....5.....6.....7.....
C
C  ---------- S I M P L E R   A L G O R I T H M ---------
C
C              T U R B U L E N T   F L O W
C            C O N S T A N T     D E N S I T Y
C.....
C
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
      LOGICAL LSTOP
      COMMON/CNTL/LSTOP
C
      OPEN(UNIT=15,FILE='INPUT',STATUS='OLD')
      OPEN(UNIT=16,FILE='OUTP',STATUS='OLD')
      OPEN(UNIT=17,FILE='PLOT15',STATUS='OLD')
C
      CALL INPUT
      CALL GRID
      CALL SETUP1
      CALL START
   10 CALL DENSE
      CALL BOUND
      CALL OUTPUT
      IF(LSTOP) THEN
      CLOSE(UNIT=15,STATUS='KEEP')
      CLOSE(UNIT=16,STATUS='KEEP')
      CLOSE(UNIT=17,STATUS='KEEP')
      STOP
      ENDIF
      CALL SETUP2
      GO TO 10
      END
C
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C
      SUBROUTINE DIFLOW
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
C
      COMMON/COEF/FLOW,DIFF,ACOF
C
C
      TOL = 1.0D-20
      ACOF = DIFF
      IF (DABS(FLOW).LT.TOL) RETURN
      TEMP=DIFF-ABS(FLOW)*0.1
      ACOF=0.
      IF(TEMP.LE.0.) RETURN
      TEMP=TEMP/DIFF
      ACOF=DIFF*TEMP**5
      RETURN
      END
C
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C
      SUBROUTINE SOLVE
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
C
      LOGICAL  LSOLVE,LBLK
```

```
      PARAMETER (ID=151,JD=151,NFD=7,NFP3=7,MIJ=151)
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     1 AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     2 X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     3 Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     4 YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     5 R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     1 FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     1 IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
      COMMON/LOGI/LSOLVE(NFP3),LBLK(NFP3)
      COMMON/VARI/TIME,DT,RELAX(NFP3),RHOCON
      COMMON/INOUT/IN,IO
      DIMENSION D(MIJ),VAR(MIJ),VARM(MIJ),VARP(MIJ),PHIBAR(MIJ)
C
      ISTF=IST-1
      JSTF=JST-1
      IT1=L2+IST
      IT2=L3+IST
      JT1=M2+JST
      JT2=M3+JST
C-----------------------------------------------------------------------
      DO 999 NT=1,NTIMES(NF)
      DO 391 N=NF,NF
      IF(.NOT.LBLK(NF))GO TO 60
C
C
C         COME HERE TO DO BLOCK CORRECTION
C
C.....................
C SUMMING IN I DIRECTION
C.....................
      DO 22 J=JST,M2
      VAR(J)=0.
      VARP(J)=0.
      VARM(J)=0.
      D(J)=0.
      DO 33 I=IST,L2
      VAR(J)=VAR(J)+AP(I,J)
      IF(I.NE.IST) VAR(J)=VAR(J)-AIM(I,J)
      IF(I.NE.L2) VAR(J )=VAR(J)-AIP(I,J)
      VARM(J)=VARM(J)+AJM(I,J)
      VARP(J)=VARP(J)+AJP(I,J)
      D(J)=D(J)+CON(I,J)+AIP(I,J)*F(I+1,J,N)+AIM(I,J)*
     1F(I-1,J,N)+AJP(I,J)*F(I,J+1,N)+AJM(I,J)*F(I,J-1,N)-
     2AP(I,J)*F(I,J,N)
   33 CONTINUE
   22 CONTINUE
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VAR(4)=1.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VARP(4)=0.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VARM(4)=0.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) D(4)=0.
      PHIBAR(M1)=0.
      PHIBAR(JSTF)=0.
      PT(JSTF)=0.
      QT(JSTF)=PHIBAR(JSTF)
      DO 44 J=JST,M2
      DENOM=VAR(J)-PT(J-1)*VARM(J)
      PT(J)=VARP(J)/DENOM
      TEMP=D(J)
      QT(J)=(TEMP+VARM(J)*QT(J-1))/DENOM
   44 CONTINUE
      DO 45 JJ=JST,M2
      J=JT1-JJ
   45 PHIBAR(J)=PHIBAR(J+1)*PT(J)+QT(J)
```

```
      DO 47 I=IST,L2
      DO 47 J=JST,M2
   47 F(I,J,N)=F(I,J,N)+PHIBAR(J)
C.....................
C SUMMING IN J DIRECTION
C.....................
      DO 51 I=IST,L2
      VAR(I)=0.
      VARP(I)=0.
      VARM(I)=0.
      D(I)=0.
      DO 53 J=JST,M2
      VAR(I)=VAR(I)+AP(I,J)
      IF(J.NE.JST) VAR(I)=VAR(I)-AJM(I,J)
      IF(J.NE.M2) VAR(I)=VAR(I)-AJP(I,J)
      VARP(I)=VARP(I)+AIP(I,J)
      VARM(I)=VARM(I)+AIM(I,J)
      D(I)=D(I)+CON(I,J)+AIP(I,J)*F(I+1,J,N)+
     1AIM(I,J)*F(I-1,J,N)+AJP(I,J)*F(I,J+1,N)+AJM(I,J)*
     2F(I,J-1,N)-AP(I,J)*F(I,J,N)
   53 CONTINUE
   51 CONTINUE
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VAR(4)=1.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VARP(4)=0.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) VARM(4)=0.
      IF((NF.EQ.3).OR.(NF.EQ.NP)) D(4)=0.
      PHIBAR(L1)=0.
      PHIBAR(ISTF)=0.
      PT(ISTF)=0.
      QT(ISTF)=PHIBAR(ISTF)
      DO 57 I=IST,L2
      DENOM=VAR(I)-PT(I-1)*VARM(I)
      PT(I)=VARP(I)/DENOM
      TEMP=D(I)
      QT(I)=(TEMP+QT(I-1)*VARM(I))/DENOM
   57 CONTINUE
      DO 58 II=IST,L2
      I=IT1-II
   58 PHIBAR(I)=PHIBAR(I+1)*PT(I)+QT(I)
      DO 59 I=IST,L2
      DO 59 J=JST,M2
   59 F(I,J,N)=F(I,J,N)+PHIBAR(I)
C-----------------------------------------------------------------
   60 CONTINUE
      DO 90 J=JST,M2
      PT(ISTF)=0.
      QT(ISTF)=F(ISTF,J,N)
      DO 70 I=IST,L2
      DENOM=AP(I,J)-PT(I-1)*AIM(I,J)
c
      testd=abs(denom)
      if(testd.lt.1.e-50)write(io,6979) nf,i,j,ap(i,j)
6979  format(2x,'solve, nf,i,j,ap ',3i5,e10.4)
c
      PT(I)=AIP(I,J)/DENOM
      TEMP=CON(I,J)+AJP(I,J)*F(I,J+1,N)+AJM(I,J)*F(I,J-1,N)
      QT(I)=(TEMP+AIM(I,J)*QT(I-1))/DENOM
   70 CONTINUE
      DO 80 II=IST,L2
      I=IT1-II
   80 F(I,J,N)=F(I+1,J,N)*PT(I)+QT(I)
   90 CONTINUE
C-----------------------------------------------------------------
      DO 190 JJ=JST,M3
      J=JT2-JJ
      PT(ISTF)=0.
```

```
      QT(ISTF)=F(ISTF,J,N)
      DO 170 I=IST,L2
      DENOM=AP(I,J)-PT(I-1)*AIM(I,J)
      PT(I)=AIP(I,J)/DENOM
      TEMP=CON(I,J)+AJP(I,J)*F(I,J+1,N)+AJM(I,J)*F(I,J-1,N)
      QT(I)=(TEMP+AIM(I,J)*QT(I-1))/DENOM
  170 CONTINUE
      DO 180 II=IST,L2
      I=IT1-II
  180 F(I,J,N)=F(I+1,J,N)*PT(I)+QT(I)
  190 CONTINUE
C-----------------------------------------------------------------
      DO 290 I=IST,L2
      PT(JSTF)=0.
      QT(JSTF)=F(I,JSTF,N)
      DO 270 J=JST,M2
      DENOM=AP(I,J)-PT(J-1)*AJM(I,J)
      PT(J)=AJP(I,J)/DENOM
      TEMP=CON(I,J)+AIP(I,J)*F(I+1,J,N)+AIM(I,J)*F(I-1,J,N)
      QT(J)=(TEMP+AJM(I,J)*QT(J-1))/DENOM
  270 CONTINUE
      DO 280 JJ=JST,M2
      J=JT1-JJ
  280 F(I,J,N)=F(I,J+1,N)*PT(J)+QT(J)
  290 CONTINUE
C-----------------------------------------------------------------
      DO 390 II=IST,L3
      I=IT2-II
      PT(JSTF)=0.
      QT(JSTF)=F(I,JSTF,N)
      DO 370 J=JST,M2
      DENOM=AP(I,J)-PT(J-1)*AJM(I,J)
      PT(J)=AJP(I,J)/DENOM
      TEMP=CON(I,J)+AIP(I,J)*F(I+1,J,N)+AIM(I,J)*F(I-1,J,N)
      QT(J)=(TEMP+AJM(I,J)*QT(J-1))/DENOM
  370 CONTINUE
      DO 380 JJ=JST,M2
      J=JT1-JJ
  380 F(I,J,N)=F(I,J+1,N)*PT(J)+QT(J)
  390 CONTINUE
  391 CONTINUE
C
  999 CONTINUE
      RETURN
C
C-----------------------------------------------------------------
C
C---------------
      ENTRY RESET
C---------------
C
      DO 400 J=2,M2
      DO 400 I=2,L2
      CON(I,J)=0.
      AP(I,J)=0.
  400 CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C
      SUBROUTINE SETUP
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
```

```
      PARAMETER (ID=151,JD=151,NFD=7,NFP3=7,MIJ=151)
C
      LOGICAL  LSOLVE,LBLK,LSTOP
C
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     1 AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     2 X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     3 Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     4 YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     5 R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
C
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     1 FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
C
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     1 IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
C
      COMMON/LOGI/LSOLVE(NFP3),LBLK(NFP3)
C
      COMMON/VARI/TIME,DT,RELAX(NFP3),RHOCON
C
      COMMON/CNTL/LSTOP
C
      COMMON/SORC/SMAX,SSUM
C
      COMMON/COEF/FLOW,DIFF,ACOF
C
      COMMON/INOUT/IN,IO
C
      DIMENSION U(ID,JD),V(ID,JD),PC(ID,JD),P(ID,JD)
C
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
     :            (F(1,1,3),PC(1,1)),(F(1,1,4),P(1,1))
C
      DIMENSION CONS(ID,JD,2),AIMS(ID,JD,3),AIPS(ID,JD,3),
     :          AJMS(ID,JD,3),AJPS(ID,JD,3),APS(ID,JD,3)
C
    1 FORMAT(15X,'COMPUTATION  IN  CARTESIAN  COORDINATES')
    2 FORMAT(15X,'COMPUTATION FOR AXISYMMETRIC SITUATION')
    3 FORMAT(15X,'COMPUTATION   IN   POLAR   COORDINATES')
    4 FORMAT(14X,40(1H*),//)
C
C
C-----------------------------------------------------------------
C
C-----------------
      ENTRY SETUP1
C-----------------
      NFMAX=NFD
      NRHO=NFD+1
      NGAM=NFD+2
      L2=L1-1
      L3=L2-1
      M2=M1-1
      M3=M2-1
      X(1)=XU(2)
      DO 5 I=2,L2
    5 X(I)=0.5*(XU(I+1)+XU(I))
      X(L1)=XU(L1)
      Y(1)=YV(2)
      DO 10 J=2,M2
   10 Y(J)=0.5*(YV(J+1)+YV(J))
      Y(M1)=YV(M1)
      DO 15 I=2,L1
   15 XDIF(I)=X(I)-X(I-1)
      DO 18 I=2,L2
```

```
18 XCV(I)=XU(I+1)-XU(I)
   DO 20 I=3,L2
20 XCVS(I)=XDIF(I)
   XCVS(3)=XCVS(3)+XDIF(2)
   XCVS(L2)=XCVS(L2)+XDIF(L1)
   DO 22 I=3,L3
   XCVI(I)=0.5*XCV(I)
22 XCVIP(I)=XCVI(I)
   XCVIP(2)=XCV(2)
   XCVI(L2)=XCV(L2)
   DO 35 J=2,M1
35 YDIF(J)=Y(J)-Y(J-1)
   DO 40 J=2,M2
40 YCV(J)=YV(J+1)-YV(J)
   DO 45 J=3,M2
45 YCVS(J)=YDIF(J)
   YCVS(3)=YCVS(3)+YDIF(2)
   YCVS(M2)=YCVS(M2)+YDIF(M1)
   IF(MODE.NE.1) GO TO 55
   DO 52 J=1,M1
   RMN(J)=1.0
52 R(J)=1.0
   GO TO 56
55 DO 50 J=2,M1
50 R(J)=R(J-1)+YDIF(J)
   RMN(2)=R(1)
   DO 60 J=3,M2
60 RMN(J)=RMN(J-1)+YCV(J-1)
   RMN(M1)=R(M1)
56 CONTINUE
   DO 57 J=1,M1
   SX(J)=1.
   SXMN(J)=1.
   IF(MODE.NE.3) GO TO 57
   SX(J)=R(J)
   IF(J.NE.1) SXMN(J)=RMN(J)
57 CONTINUE
   DO 62 J=2,M2
   YCVR(J)=R(J)*YCV(J)
   ARX(J)=YCVR(J)
   IF(MODE.NE.3) GO TO 62
   ARX(J)=YCV(J)
62 CONTINUE
   DO 64 J=4,M3
64 YCVRS(J)=0.5*(R(J)+R(J-1))*YDIF(J)
   YCVRS(3)=0.5*(R(3)+R(1))*YCVS(3)
   YCVRS(M2)=0.5*(R(M1)+R(M3))*YCVS(M2)
   IF(MODE.NE.2) GO TO 67
   DO 65 J=3,M3
   ARXJ(J)=0.25*(1.+RMN(J)/R(J))*ARX(J)
65 ARXJP(J)=ARX(J)-ARXJ(J)
   GO TO 68
67 DO 66 J=3,M3
   ARXJ(J)=0.5*ARX(J)
66 ARXJP(J)=ARXJ(J)
68 ARXJP(2)=ARX(2)
   ARXJ(M2)=ARX(M2)
   DO 70 J=3,M3
   FV(J)=ARXJP(J)/ARX(J)
70 FVP(J)=1.-FV(J)
   DO 85 I=3,L2
   FX(I)=0.5*XCV(I-1)/XDIF(I)
85 FXM(I)=1.-FX(I)
   FX(2)=0.
   FXM(2)=1.
   FX(L1)=1.
```

```
      FXM(L1)=0.
      DO 90 J=3,M2
      FY(J)=0.5*YCV(J-1)/YDIF(J)
   90 FYM(J)=1.-FY(J)
      FY(2)=0.
      FYM(2)=1.
      FY(M1)=1.
      FYM(M1)=0.
C-----------------------------------------------------------------
C        CON,AP,U,V,RHO,PC AND P ARRAYS ARE INITIALIZED HERE
C-----------------------------------------------------------------
      DO 95 J=1,M1
      DO 95 I=1,L1
      PC(I,J)=0.
      U(I,J)=0.
      V(I,J)=0.
      CON(I,J)=0.
      AP(I,J)=0.
      P(I,J)=0.
   95 CONTINUE
      RHO = RHOCON
      IF(MODE.EQ.1) WRITE(IO,1)
      IF(MODE.EQ.2) WRITE(IO,2)
      IF(MODE.EQ.3) WRITE(IO,3)
      WRITE(IO,4)
      RETURN
C
C-----------------------------------------------------------------
C
C-----------------
      ENTRY SETUP2
C-----------------
C
C     COEFFICIENTS FOR THE U EQUATION
C===============
      CALL RESET
C===============
      NF=1
      IF(.NOT.LSOLVE(NF)) GO TO 100
      IST=3
      JST=2
C===============
      CALL GAMSOR
C===============
      REL=1.-RELAX(NF)
      DO 102 I=3,L2
      FL=XCVI(I)*V(I,2)
      FLM=XCVIP(I-1)*V(I-1,2)
      FLOW=R(1)*(FL+FLM)*RHO
      DIFF=R(1)*(XCVI(I)*GAM(I,1)+XCVIP(I-1)*GAM(I-1,1))/YDIF(2)
C===============
      CALL DIFLOW
C===============
  102 AJM(I,2)=ACOF+DMAX1(0.0D0,FLOW)
      DO 103 J=2,M2
      FLOW=ARX(J)*U(2,J)*RHO
      DIFF=ARX(J)*GAM(1,J)/(XCV(2)*SX(J))
C===============
      CALL DIFLOW
C===============
      AIM(3,J)=ACOF+DMAX1(0.0D0,FLOW)
      DO 103 I=3,L2
      IF(I.EQ.L2) GO TO 104
      FL=U(I,J)*RHO
      FLP=U(I+1,J)*RHO
      FLOW=ARX(J)*0.5*(FL+FLP)
```

```
      DIFF=ARX(J)*GAM(I,J)/(XCV(I)*SX(J))
      GO TO 105
  104 FLOW=ARX(J)*U(L1,J)*RHO
      DIFF=ARX(J)*GAM(L1,J)/(XCV(L2)*SX(J))
C===============
  105 CALL DIFLOW
C===============
      AIM(I+1,J)=ACOF+DMAX1(0.0D0,FLOW)
      AIP(I,J)=AIM(I+1,J)-FLOW
      IF(J.EQ.M2) GO TO 106
      FL=XCVI(I)*V(I,J+1)*RHO
      FLM=XCVIP(I-1)*V(I-1,J+1)*RHO
      GM=GAM(I,J)*GAM(I,J+1)/(YCV(J)*GAM(I,J+1)+YCV(J+1)*GAM(I,J)+
     1 1.0D-30)*XCVI(I)
      GMM=GAM(I-1,J)*GAM(I-1,J+1)/(YCV(J)*GAM(I-1,J+1)+YCV(J+1)*
     1 GAM(I-1,J)+1.D-30)*XCVIP(I-1)
      DIFF=RMN(J+1)*2.*(GM+GMM)
      GO TO 107
  106 FL=XCVI(I)*V(I,M1)*RHO
      FLM=XCVIP(I-1)*V(I-1,M1)*RHO
      DIFF=R(M1)*(XCVI(I)*GAM(I,M1)+XCVIP(I-1)*GAM(I-1,M1))/YDIF(M1)
  107 FLOW=RMN(J+1)*(FL+FLM)
C===============
      CALL DIFLOW
C===============
      AJM(I,J+1)=ACOF+DMAX1(0.0D0,FLOW)
      AJP(I,J)=AJM(I,J+1)-FLOW
      VOL=YCVR(J)*XCVS(I)
      APT=RHO*(XCVI(I)+XCVIP(I-1))
     1/(XCVS(I)*DT)
      AP(I,J)=AP(I,J)-APT
      CON(I,J)=CON(I,J)+APT*U(I,J)
C==============
      CALL WALL
C==============
      AP(I,J)=(-AP(I,J)*VOL+AIP(I,J)+AIM(I,J)+AJP(I,J)+AJM(I,J))
     1/RELAX(NF)
      CON(I,J)=CON(I,J)*VOL+REL*AP(I,J)*U(I,J)
      DU(I,J)=VOL/(XDIF(I)*SX(J))
      DU(I,J)=DU(I,J)/AP(I,J)
c
      testap=abs(ap(i,j))
      if(testap.lt.0.1e-50) write(io,6969)nf,i,j,ap(i,j)
6969  format(2x,'nf,i,j,ap ',3i5,e10.4)
c
  103 CONTINUE
C
CCCC
```

```
     1AJM(I,J)*U(I,J-1)+CON(I,J))/AP(I,J)
  100 CONTINUE
C
C     COEFFICIENTS FOR THE  V  EQUATION
C
C=============
      CALL RESET
C=============
      NF=2
      IF(.NOT.LSOLVE(NF)) GO TO 200
      IST=2
      JST=3
C=============
      CALL GAMSOR
C=============
      REL=1.-RELAX(NF)
      DO 202 I=2,L2
      AREA=R(1)*XCV(I)
      FLOW=AREA*V(I,2)*RHO
      DIFF=AREA*GAM(I,1)/YCV(2)
C=============
      CALL DIFLOW
C=============
  202 AJM(I,3)=ACOF+DMAX1(0.0D0,FLOW)
      DO 203 J=3,M2
      FL=ARXJ(J)*U(2,J)*RHO
      FLM=ARXJP(J-1)*U(2,J-1)*RHO
      FLOW=FL+FLM
      DIFF=(ARXJ(J)*GAM(1,J)+ARXJP(J-1)*GAM(1,J-1))/(XDIF(2)*SXMN(J))
C=============
      CALL DIFLOW
C=============
      AIM(2,J)=ACOF+DMAX1(0.0D0,FLOW)
      DO 203 I=2,L2
      IF(I.EQ.L2) GO TO 204
      FL=ARXJ(J)*U(I+1,J)*RHO
      FLM=ARXJP(J-1)*U(I+1,J-1)*RHO
      GM=GAM(I,J)*GAM(I+1,J)/(XCV(I)*GAM(I+1,J)+XCV(I+1)*GAM(I,J)+
     1 1.D-30)*ARXJ(J)
      GMM=GAM(I,J-1)*GAM(I+1,J-1)/(XCV(I)*GAM(I+1,J-1)+XCV(I+1)*
     1 GAM(I,J-1)+1.0D-30)*ARXJP(J-1)
      DIFF=2.*(GM+GMM)/SXMN(J)
      GO TO 205
  204 FL=ARXJ(J)*U(L1,J)*RHO
      FLM=ARXJP(J-1)*U(L1,J-1)*RHO
      DIFF=(ARXJ(J)*GAM(L1,J)+ARXJP(J-1)*GAM(L1,J-1))/(XDIF(L1)*SXMN(J))
  205 FLOW=FL+FLM
C=============
      CALL DIFLOW
C=============
      AIM(I+1,J)=ACOF+DMAX1(0.0D0,FLOW)
      AIP(I,J)=AIM(I+1,J)-FLOW
      IF(J.EQ.M2) GO TO 206
      AREA=R(J)*XCV(I)
      FL=V(I,J)*RHO*RMN(J)
      FLP=V(I,J+1)*RHO*RMN(J+1)
      FLOW=(FV(J)*FL+FVP(J)*FLP)*XCV(I)
      DIFF=AREA*GAM(I,J)/YCV(J)
      GO TO 207
  206 AREA=R(M1)*XCV(I)
      FLOW=AREA*V(I,M1)*RHO
      DIFF=AREA*GAM(I,M1)/YCV(M2)
C=============
  207 CALL DIFLOW
C=============
      AJM(I,J+1)=ACOF+DMAX1(0.0D0,FLOW)
```

```
      AJP(I,J)=AJM(I,J+1)-FLOW
      VOL=YCVRS(J)*XCV(I)
      APT=(ARXJ(J)*RHO*0.5*(SX(J)+SXMN(J))+ARXJP(J-1)*RHO*
     10.5*(SX(J-1)+SXMN(J)))/(YCVRS(J)*DT)
      AP(I,J)=AP(I,J)-APT
      CON(I,J)=CON(I,J)+APT*V(I,J)
C=============
      CALL WALL
C=============
      AP(I,J)=(-AP(I,J)*VOL+AIP(I,J)+AIM(I,J)+AJP(I,J)+AJM(I,J))
     1/RELAX(NF)
c
      testap=abs(ap(i,j))
      if(testap.lt.0.1e-50) write(io,6969)nf,i,j,ap(i,j)
c
      CON(I,J)=CON(I,J)*VOL+REL*AP(I,J)*V(I,J)
      DV(I,J)=VOL/YDIF(J)
      DV(I,J)=DV(I,J)/AP(I,J)
  203 CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 209 J = 1,M1
      DO 209 I = 1,L1
      AIPS(I,J,2) = AIP(I,J)
      AIMS(I,J,2) = AIM(I,J)
      AJPS(I,J,2) = AJP(I,J)
      AJMS(I,J,2) = AJM(I,J)
      APS(I,J,2) = AP(I,J)
      CONS(I,J,2) = CON(I,J)
 209  CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
  200 CONTINUE
C
C     COEFFICIENTS FOR THE PRESSURE EQUATION
C
      NF=NP
      IF(.NOT.LSOLVE(NF)) GO TO 500
      IST=2
      JST=2
      DO 402 I=2,L2
      ARHO=R(1)*XCV(I)*RHO
      CON(I,2)=ARHO*V(I,2)
  402 AJM(I,2)=0.
      DO 403 J=2,M2
      ARHO=ARX(J)*RHO
      CON(2,J)=CON(2,J)+ARHO*U(2,J)
      AIM(2,J)=0.
      DO 403 I=2,L2
      IF(I.EQ.L2) GO TO 404
      ARHO=ARX(J)*RHO
      FLOW=ARHO*PC(I+1,J)
      CON(I,J)=CON(I,J)-FLOW
      CON(I+1,J)=CON(I+1,J)+FLOW
      AIP(I,J)=ARHO*DU(I+1,J)
      AIM(I+1,J)=AIP(I,J)
      GO TO 405
  404 ARHO=ARX(J)*RHO
      CON(I,J)=CON(I,J)-ARHO*U(L1,J)
      AIP(I,J)=0.
  405 IF(J.EQ.M2) GO TO 406
      ARHO=RMN(J+1)*XCV(I)*RHO
      VHAT=(AIP(I,J+1)*V(I+1,J+1)+AIM(I,J+1)*V(I-1,J+1)+AJP(I,J+1)*
     1V(I,J+2)+AJM(I,J+1)*V(I,J)+CON(I,J+1))/AP(I,J+1)
```

```
      FLOW=ARHO*VHAT
      CON(I,J)=CON(I,J)-FLOW
      CON(I,J+1)=FLOW
      AJP(I,J)=ARHO*DV(I,J+1)
      AJM(I,J+1)=AJP(I,J)
      GO TO 407
  406 ARHO=RMN(M1)*XCV(I)*RHO
      CON(I,J)=CON(I,J)-ARHO*V(I,M1)
      AJP(I,J)=0.
  407 AP(I,J)=AIP(I,J)+AIM(I,J)+AJP(I,J)+AJM(I,J)
  403 CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 409 J = 1,M1
      DO 409 I = 1,L1
      AIPS(I,J,3) = AIP(I,J)
      AIMS(I,J,3) = AIM(I,J)
      AJPS(I,J,3) = AJP(I,J)
      AJMS(I,J,3) = AJM(I,J)
      APS(I,J,3) = AP(I,J)
 409  CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IF(ITER.LE.1) GO TO 411
      DO 410 J=2,M2
      DO 410 I=2,L2
      AP(I,J)=AP(I,J)/RELAX(NP)
      CON(I,J)=CON(I,J)+(1.-RELAX(NP))*AP(I,J)*P(I,J)
  410 CONTINUE
  411 CONTINUE
C===============
      CALL SOLVE
C===============
      NF=1
      IST=3
      JST=2
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 412 J = 1,M1
      DO 412 I = 1,L1
      CON(I,J) = CONS(I,J,1)
      AIP(I,J) = AIPS(I,J,1)
      AIM(I,J) = AIMS(I,J,1)
      AJP(I,J) = AJPS(I,J,1)
      AJM(I,J) = AJMS(I,J,1)
      AP(I,J) = APS(I,J,1)
 412  CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 413 J=2,M2
      DO 413 I=3,L2
  413 CON(I,J)=CON(I,J)+DU(I,J)*AP(I,J)*(P(I-1,J)-P(I,J))
C===============
      CALL SOLVE
C===============
      NF=2
      IST=2
      JST=3
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 420 J = 1,M1
```

```
      DO 420 I = 1,L1
      CON(I,J) = CONS(I,J,2)
      AIP(I,J) = AIPS(I,J,2)
      AIM(I,J) = AIMS(I,J,2)
      AJP(I,J) = AJPS(I,J,2)
      AJM(I,J) = AJMS(I,J,2)
      AP(I,J) = APS(I,J,2)
420   CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 424 J=3,M2
      DO 424 I=2,L2
  424 CON(I,J)=CON(I,J)+DV(I,J)*AP(I,J)*(P(I,J-1)-P(I,J))
C==============
      CALL SOLVE
C==============
C
C     COEFFICIENTS FOR THE PRESSURE CORRECTION EQUATION
C
C==============
      CALL RESET
C==============
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DO 430 J = 1,M1
      DO 430 I = 1,L1
      AIP(I,J) = AIPS(I,J,3)
      AIM(I,J) = AIMS(I,J,3)
      AJP(I,J) = AJPS(I,J,3)
      AJM(I,J) = AJMS(I,J,3)
      AP(I,J) = APS(I,J,3)
 430  CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      NF=3
      IF(.NOT.LSOLVE(NF)) GO TO 500
      IST=2
      JST=2
C==============
      CALL GAMSOR
C==============
      SMAX=0.
      SSUM=0.
      DO 460 J=2,M2
      DO 460 I=2,L2
      VOL=YCVR(J)*XCV(I)
  460 CON(I,J)=CON(I,J)*VOL
      DO 474 I=2,L2
      ARHO=R(1)*XCV(I)*RHO
  474 CON(I,2)=CON(I,2)+ARHO*V(I,2)
      DO 475 J=2,M2
      ARHO=ARX(J)*RHO
      CON(2,J)=CON(2,J)+ARHO*U(2,J)
      DO 475 I=2,L2
      IF(I.EQ.L2) GO TO 476
      ARHO=ARX(J)*RHO
      FLOW=ARHO*U(I+1,J)
      CON(I,J)=CON(I,J)-FLOW
      CON(I+1,J)=CON(I+1,J)+FLOW
      GO TO 477
  476 ARHO=ARX(J)*RHO
      CON(I,J)=CON(I,J)-ARHO*U(L1,J)
  477 IF(J.EQ.M2) GO TO 478
```

```
      ARHO=RMN(J+1)*XCV(I)*RHO
      FLOW=ARHO*V(I,J+1)
      CON(I,J)=CON(I,J)-FLOW
      CON(I,J+1)=CON(I,J+1)+FLOW
      GO TO 479
  478 ARHO=RMN(M1)*XCV(I)*RHO
      CON(I,J)=CON(I,J)-ARHO*V(I,M1)
  479 PC(I,J)=0.
      SMAX=DMAX1(SMAX,ABS(CON(I,J)))
      SSUM=SSUM+CON(I,J)
  475 CONTINUE
C=============
      CALL SOLVE
C=============
C
C     COME HERE TO CORRECT THE VELOCITIES
C
      DO 501 J=2,M2
      DO 501 I=2,L2
      IF(I.NE.2) U(I,J)=U(I,J)+DU(I,J)*(PC(I-1,J)-PC(I,J))
      IF(J.NE.2) V(I,J)=V(I,J)+DV(I,J)*(PC(I,J-1)-PC(I,J))
  501 CONTINUE
  500 CONTINUE
C=============
      CALL GRADIENT
C=============
C
C     COEFFICIENTS FOR OTHER EQUATIONS
C
      IST=2
      JST=2
      DO 600 NF=NFMIN,NFMAX
      IF(.NOT.LSOLVE(NF)) GO TO 600
C=============
      CALL RESET
C=============
      CALL GAMSOR
C=============
      REL=1.-RELAX(NF)
      DO 602 I=2,L2
      AREA=R(1)*XCV(I)
      FLOW=AREA*V(I,2)*RHO
      DIFF=AREA*GAM(I,1)/YDIF(2)
C=============
      CALL DIFLOW
C=============
  602 AJM(I,2)=ACOF+DMAX1(0.0D0,FLOW)
      DO 603 J=2,M2
      FLOW=ARX(J)*U(2,J)*RHO
      DIFF=ARX(J)*GAM(1,J)/(XDIF(2)*SX(J))
C=============
      CALL DIFLOW
C=============
      AIM(2,J)=ACOF+DMAX1(0.0D0,FLOW)
      DO 603 I=2,L2
      IF(I.EQ.L2) GO TO 604
      FLOW=ARX(J)*U(I+1,J)*RHO
      DIFF=ARX(J)*2.*GAM(I,J)*GAM(I+1,J)/((XCV(I)*GAM(I+1,J)+
     1 XCV(I+1)*GAM(I,J)+1.0D-30)*SX(J))
      GO TO 605
  604 FLOW=ARX(J)*U(L1,J)*RHO
      DIFF=ARX(J)*GAM(L1,J)/(XDIF(L1)*SX(J))
C=============
  605 CALL DIFLOW
C=============
      AIM(I+1,J)=ACOF+DMAX1(0.0D0,FLOW)
```

```
      AIP(I,J)=AIM(I+1,J)-FLOW
      AREA=RMN(J+1)*XCV(I)
      IF(J.EQ.M2) GO TO 606
      FLOW=AREA*V(I,J+1)*RHO
      DIFF=AREA*2.*GAM(I,J)*GAM(I,J+1)/(YCV(J)*GAM(I,J+1)+
     1 YCV(J+1)*GAM(I,J)+1.0D-30)
      GO TO 607
  606 FLOW=AREA*V(I,M1)*RHO
      DIFF=AREA*GAM(I,M1)/YDIF(M1)
C=================
  607 CALL DIFLOW
C=================
      AJM(I,J+1)=ACOF+DMAX1(0.0D0,FLOW)
      AJP(I,J)=AJM(I,J+1)-FLOW
      VOL=YCVR(J)*XCV(I)
      APT=RHO/DT
      AP(I,J)=AP(I,J)-APT
      CON(I,J)=CON(I,J)+APT*F(I,J,NF)
C=================
      CALL WALL
C=================
      AP(I,J)=(-AP(I,J)*VOL+AIP(I,J)+AIM(I,J)+AJP(I,J)+AJM(I,J))
     1/RELAX(NF)
c
      testap=abs(ap(i,j))
      if(testap.lt.0.1e-50) write(io,6969)nf,i,j,ap(i,j)
c
      CON(I,J)=CON(I,J)*VOL+REL*AP(I,J)*F(I,J,NF)
  603 CONTINUE
C=================
      CALL SOLVE
C=================
  600 CONTINUE
      TIME=TIME+DT
      ITER=ITER+1
      IF(ITER.GE.LAST) LSTOP=.TRUE.
      RETURN
      END
```

APPENDIX B
User Routine for Pipe Flow

The USER routine for turbulent pipe flow is given in this appendix along with the results for two problems. In this routine much of the data is defined in a BLOCK DATA section of the code. The output control is included in the SUBROUTINE USER and not in a separate routine. A few of the input variables need explanation. For several of the cases run the inlet velocity was forced to be a power law distribution that mimicked the empirically determined distributions reported in the literature for fully developed turbulent flow. The power in the power law distribution is EN in the code. If IVEL = 1 then the power law distribution is invoked. If it = 0 then a top hat velocity distribution is used at the inlet. For the first ten iterations the velocity field is everywhere set equal to the inlet distribution. The grid is nonuniform in the radial (y) direction, the spacing being exponential. PWR is the power in the exponential expression for the grid spacing. The parameter LAST is the total number of iterations to be performed. The rest of the input is self–explanatory and the listing has many comment statements for explanation of each section of the routine.

The first problem for which output is given is CASE 1 in the report. The grid has 42 nodes axially, 26 radially. An exponential grid spacing with PWR = 4.4 is used and a power law velocity distribution at the inlet with EN = 9.0 is used. One thousand iterations were performed in solving the problem. The validity of the grid spacing is checked by looking at the value of Y^+ for points adjacent to the pipe wall. As can be seen in the output file, it is well below the upper limit of 30. The parameter DIST is the theoretical distance a node can be from the wall and still have Y^+ greater than 11.5.

The other problem output given is for Case 3. Ten grid nodes are used in each direction with the velocity grids being equally spaced. A top hat inlet velocity distribution was used. Five hundred iterations were performed to obtain the solution. This is an exact replica of the model used by Pun and Spalding [5].

```
C
C.....................................................................
C
C            ------   T U B E   V E L O C I T Y   ------
C
C                     T U R B U L E N T   F L O W
C
C.....................................................................
C
      BLOCK DATA
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TITLE*2
      LOGICAL  LSOLVE,LPRINT,LBLK,LSTOP,LPLOT,LBUG
      PARAMETER (ID=101,JD=101,NFD=6,NFP3=6,MIJ=101)
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     1 AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     2 X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     3 Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     4 YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     5 R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     1 FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     1 IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
      COMMON/LOGI/LSOLVE(NFP3),LPRINT(NFP3),LPLOT(NFP3),LBLK(NFP3)
      COMMON/VARI/TIME,DT,XL,YL,RELAX(NFP3),RHOCON
      COMMON/TTLE/TITLE(NFP3)
      COMMON/CNTL/LSTOP
      COMMON/SORC/SMAX,SSUM
      COMMON/COEF/FLOW,DIFF,ACOF
      COMMON/BUG/LBUG
      COMMON/INOUT/IN,IO
      DIMENSION U(ID,JD),V(ID,JD),PC(ID,JD),P(ID,JD),TKE(ID,JD),
     :          TED(ID,JD)
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
     : (F(1,1,3),PC(1,1)),(F(1,1,4),P(1,1)),(F(1,1,5),TKE(1,1)),
     : (F(1,1,6),TED(1,1))
C
C         F(I,J,1) = U
C         F(I,J,2) = V
C         F(I,J,3) = PC
C         F(I,J,4) = P
C         F(I,J,5) = TKE
C         F(I,J,6) = TED
C
C
      DATA IN,IO/15,16/
      DATA TITLE/'U','V','PC','P','KE','ED'/
      DATA NFMIN,NFMAX/5,6/
      DATA NU,NV,NPC,NP,NKE,NED/1,2,3,4,5,6/
C
      DATA LSOLVE(1 ),LPRINT(1 ),LPLOT(1 )/.true. ,.FALSE.,.FALSE./
      DATA LSOLVE(2 ),LPRINT(2 ),LPLOT(2 )/.true. ,.FALSE.,.FALSE./
      DATA LSOLVE(3 ),LPRINT(3 ),LPLOT(3 )/.true. ,.FALSE.,.false./
      DATA LSOLVE(4 ),LPRINT(4 ),LPLOT(4 )/.true. ,.FALSE.,.FALSE./
      DATA LSOLVE(5 ),LPRINT(5 ),LPLOT(5 )/.true. ,.false.,.false./
      DATA LSOLVE(6 ),LPRINT(6 ),LPLOT(6 )/.true. ,.FALSE.,.FALSE./
C
      DATA LBLK/NFP3*.FALSE./
      DATA LSTOP/.FALSE./
      DATA TIME,DT,ITER/0.0, 1.E+20, 0.0/
      DATA LBUG/.false./
      DATA RELAX/0.5,0.5,0.8,0.8,1.,1./
      DATA NTIMES/6*10/
C
C...PROBLEM DATA
```

```
C
      data rhocon/1.00/
C...GRID DATA
      DATA IPREF,JPREF/2,2/
      DATA MODE/2/
      END
C
C*****************************************************************************
C
      SUBROUTINE USER
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TITLE*2
      LOGICAL  LSOLVE,LPRINT,LBLK,LSTOP,LPLOT,LBUG
      PARAMETER (ID=101,JD=101,NFD=6,NFP3=6,MIJ=101)
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     1 AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     2 X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     3 Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     4 YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     5 R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     1 FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     1 IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
      COMMON/LOGI/LSOLVE(NFP3),LPRINT(NFP3),LPLOT(NFP3),LBLK(NFP3)
      COMMON/VARI/TIME,DT,XL,YL,RELAX(NFP3),RHOCON
      COMMON/TTLE/TITLE(NFP3)
      COMMON/CNTL/LSTOP
      COMMON/SORC/SMAX,SSUM
      COMMON/COEF/FLOW,DIFF,ACOF
      COMMON/BUG/LBUG
      COMMON/INOUT/IN,IO
      DIMENSION U(ID,JD),V(ID,JD),PC(ID,JD),P(ID,JD),TKE(ID,JD),
     :          TED(ID,JD)
      DIMENSION AMUT(ID,JD),TAUN(ID),PLUN(ID),GK(ID,JD)
      DIMENSION TAUNW(ID),UOUF(ID),YOYF(ID),UOUMAX(ID),YOR(ID),
     :          DELTA(ID),DY2(ID)
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
     : (F(1,1,3),PC(1,1)),(F(1,1,4),P(1,1)),(F(1,1,5),TKE(1,1)),
     : (F(1,1,6),TED(1,1))
C
C         F(I,J,1) = U
C         F(I,J,2) = V
C         F(I,J,3) = PC
C         F(I,J,4) = P
C         F(I,J,5) = TKE
C         F(I,J,6) = TED
C
C...PROBLEM DATA
C
      DATA AMU/0.0001/
C
      DATA C1,C2,CD,CAPPA,ECON,PRTKE,PRTED/ 1.43, 1.92, 0.09, 0.4, 9.0,
     :                                      1.0,1.3 /
C
      IF (LBUG) PRINT *,"USER"
C     -----------
      ENTRY GRID
C     -----------
      IF (LBUG) PRINT *, "ENTER GRID"
C
C        L1       NUMBER OF X GRID POINTS
C        M1       NUMBER OF Y GRID POINTS
C        XL       LENGTH IN AXIAL DIRECTION
C        YL       LENGTH IN RADIAL DIRECTION
C        XINC     AXIAL GRID SPACING
```

```
C         YINC     RADIAL GRID SPACING
C         REYN     INPUT REYNOLDS NUMBER
C         PWR      EXPONENT FOR EXPONENTIAL GRID SPACING
C         EN       EXPONENT FOR POWER LAW INLET VELOCITY DIST.
C         IVEL     =1 POWER LAW INPUT VELOCITY
C                  =0 TOP HAT INPUT VELOCITY
C
      READ(15,*) LAST
      READ(15,*) L1, M1, XL, YL
      READ(15,*) REYN, IVEL, PWR, EN
      UAVG = REYN*AMU/(RHOCON*2.0*YL)
      L2 = L1 - 1
      L3 = L1 - 2
      M2 = M1 - 1
C
      XINC = XL/(L1-2)
      XU(1) = 0.0
      XU(2) = 0.0
      DO 1 I = 2,L1
         XU(I+1) = XU(I) + XINC
    1    CONTINUE
C
      YV(2) = 0.0
      YYY = 0.0
      IF (PWR .LE. 1.e-20) THEN
C
C           UNIFORM GRID
C
      YINC = YL/(M1-2)
      DO 2 J = 3,M1
      YV(J) = YV(J-1) + YINC
    2 CONTINUE
      ELSE
C
C           EXPONENTIAL GRID
C
      YINC = 1.0/(M1-2)
      YCOR = YL*EXP(-PWR)
      DO 3 J=3,M1
         YYY = YYY + YINC
         YV(J) = YL*(1.0 - EXP(-PWR*YYY)) + YCOR
    3    CONTINUE
      ENDIF
C
C       TURBULENCE CONSTANTS
C
      CD5 = CD**.5
      CD25 = CD**.25
      CD75 = CD**.75
      C2M = C2 - 1.0
      TC2M = 2.*C2 - 1.0
C
      RETURN
C
C     -----------
      ENTRY START
C     -----------
      IF (LBUG) PRINT *, "ENTER START"
C
C      OPEN RESTART FILE
C
       OPEN(UNIT=12,FILE='RESTART',FORM='UNFORMATTED')
C
C
      DO 10  K = 1,NFD
       DO 10  J = 1,M1
```

```
        DO 10  I = 1,L1
           F(I,J,K) = 0.0
  10     CONTINUE
C
C      STARTING VELOCITY DISTRIBUTION
C      BASED ON THE POWER LAW
C      U/Umax = (1 - r/R)**(1/n)
C
      IF (IVEL .EQ. 0)  EN = 1.E30
      UCTR = UAVG*(1.0 + 1.5/EN + 0.5/EN**2)
C
      DO 11 J = 1,M2
       DO 11 I = 1,L1
        U(I,J) = UCTR*(1.0 - Y(J)/Y(M1))**(1./EN)
        TKE(I,J) = 0.005*U(I,J)**2
        TED(I,J) = CD*TKE(I,J)**1.5/(0.03*Y(M1))
   11    CONTINUE
C
      RETURN
C
C     -----------
      ENTRY DENSE
C     -----------
      IF (LBUG) PRINT *, "ENTER DENSE"
C
      DO 200 J = 1,M1
      DO 200 I = 1,L1
         AMUT(I,J) = CD*RHOCON*TKE(I,J)**2/(TED(I,J)+1.E-20)
  200 CONTINUE
      RETURN
C
C     -----------
      ENTRY BOUND
C     -----------
      IF (LBUG) PRINT *, "ENTER BOUND"
      IF (ITER .LT. 10) THEN
       DO 250 J = 1,M2
       DO 250 I = 1,L1
        U(I,J) = UCTR*(1.0 - Y(J)/Y(M1))**(1./EN)
  250    CONTINUE
      ENDIF
C
C...BOUNDARY CONDITIONS
C
C     U - VELOCITY
C
      DO 300 I = 2,L1
         U(I,1) = U(I,2)
         U(I,M1) = 0.0
  300    CONTINUE
      DO 302 J=1,M2
         U(2,J) = UCTR*(1.0 - Y(J)/Y(M1))**(1./EN)
         U(L1,J) = U(L2,J)
  302   CONTINUE
C
C       V - VELOCITY
C
      DO 303 I = 1,L1
         V(I,2) = 0.0
         V(I,M1) = 0.0
  303    CONTINUE
      DO 304 J = 2,M1
         V(1,J) = 0.0
         V(L1,J) = 0.0
  304    CONTINUE
C
```

```
C     P - PRESSURE  &  PRESSURE CORRECTION
C
      do 305 j = 2,m2
      p(l1,j) = (p(l2,j)*xcvs(l2)-p(l3,j)*xdif(l1))/xdif(l2)
      p(1,j) = (p(2,j)*xcvs(3)-p(3,j)*xdif(2))/xdif(3)
  305 continue
      do 306 i = 2,l2
      p(i,1) = (p(i,2)*ycvs(3)-p(i,3)*ydif(2))/ydif(3)
      p(i,m1) = (p(i,m2)*ycvs(m2)-p(i,m3)*ydif(m1))/ydif(m2)
  306 continue
      p(1,1) = p(2,1)+p(1,2)-p(2,2)
      p(l1,1) = p(l2,1)+p(l1,2)-p(l2,2)
      p(1,m1) = p(2,m1)+p(1,m2)-p(2,m2)
      p(l1,m1)= p(l2,m1)+p(l1,m2)-p(l2,m2)
C
C       TURBULENT ENERGY
C
      DO 350 J = 1,M2
        TKE(1,J) = 0.005*U(2,J)**2
        TED(1,J) = CD*TKE(1,J)**1.5/(0.03*Y(M1))
        TKE(L1,J) = TKE(L2,J)
        TED(L1,J) = TED(L2,J)
  350   CONTINUE
C
      DO 352 I = 1,L1
        TKE(I,1) = TKE(I,2)
        TED(I,1) = TED(I,2)
  352   CONTINUE
C
C       WALL STRESS
C
C       SHEAR STRESS/VELOCITY & Y+ AT OUTER RADIUS
C
      DO 360 I = 2,L2
        DIST = Y(M1) - Y(M2)
        CT = CD25*DIST
        RK = RHOCON*SQRT(ABS(TKE(I,M2)))
        PLUN(I) = RK*CT/AMU
        IF (PLUN(I) .GT. 11.5) THEN
          EPLUN = ECON*PLUN(I)
          TAUN(I) = CAPPA*RK*CD25/DLOG(EPLUN)
        ELSE
          TAUN(I) = AMU/DIST
        ENDIF
C
C          THE APPROPRIATE DISTANCE FROM THE WALL FOR
C          THE MAIN GRID NODE NEAREST THE WALL
C
        DELTA(I) = 12.0*AMU/(RK*CD25 + 1.e-20)
  360   CONTINUE
      RETURN
C       --------------
        ENTRY GRADIENT
C       --------------
C       COMPUTE GENK=GK(I,J) AT MAIN GRID POINTS FOR USE
C       IN TURBULENT ENERGY AND DISSIPATION EQUATIONS
C
      DO 402 J = 2,M2
      DO 402 I = 2,L2
      UPX = (U(I+1,J) - U(I,J))/XCV(I)
      VPY = (V(I,J+1) - V(I,J))/YCV(J)
      VOR = 0.5*(V(I,J+1) + V(I,J))/Y(J)
cccc      UPY = 0.5*(U(I+1,J+1) - U(I+1,J-1) + U(I,J+1) - U(I,J-1))
cccc    :          /(Y(J+1) - Y(J-1))
      FAC1 = (YV(J+1) - Y(J))/YDIF(J+1)
      FAC2 = (YV(J) - Y(J-1))/YDIF(J)
```

```
        UNEC = U(I,J) + (U(I,J+1) - U(I,J))*FAC1
        USEC = U(I,J-1) + (U(I,J) - U(I,J-1))*FAC2
        UNWC = U(I+1,J) + (U(I+1,J+1) - U(I+1,J))*FAC1
        USWC = U(I+1,J-1) + (U(I+1,J) - U(I+1,J-1))*FAC2
        UPY = 0.5*(UNEC - USEC + UNWC - USWC)/YCV(J)
        VPX = 0.5*(V(I+1,J+1) - V(I-1,J+1) + V(I+1,J) - V(I-1,J))
     :          /(X(I+1) - X(I-1))
        GK(I,J) = AMUT(I,J)*(2.*(UPX*UPX + VPX*VPX + VOR*VOR)
     :              + (UPY + VPX)**2)
  402  CONTINUE
       RETURN
C
C      ------------
       ENTRY GAMSOR
C      ------------
C
C          S = SC + SP*U  SO, AP = SP  &  CON = SC
C
       IF (LBUG) PRINT *, "ENTER GAMSOR"
       GO TO (510,520,530,599,550,560),NF
C
C        U - VELOCITY
C
510    CONTINUE
       DO 512 J = 1,M1
       DO 511 I = 1,L1
          GAM(I,J) = AMU + AMUT(I,J)
  511   CONTINUE
        GAM(L1,J) = 0.0
  512   CONTINUE
       DO 514 I = 2,L1
          GAM(I,1) = 0.0
  514   CONTINUE
C
       RETURN
C
C        V - VELOCITY
C
520    DO 521 J = 1,M1
       DO 521 I = 1,L1
           GAM(I,J) = AMU + AMUT(I,J)
           GAM(1,J) = 0.0
  521      CONTINUE
       RETURN
C
C        PRESSURE CORRECTION
C
C        ADJUST U TO CONSERVE MASS FLOW
C
530    CONTINUE
       QIN = 0.0
       DO 531 J = 2,M1
           DY2(J) = Y(J)**2 - Y(J-1)**2
           QIN = QIN + (U(2,J-1) + U(2,J))*DY2(J)
  531      CONTINUE
       DO 533 I = 3,L1
        QOUT = 0.0
        DO 532 J = 2,M1
            QOUT = QOUT +(U(I,J-1) + U(I,J))*DY2(J)
  532       CONTINUE
            QINOUT = QIN/QOUT
        DO 533 J = 1,M2
            U(I,J) = U(I,J)*QINOUT
  533       CONTINUE
      RETURN
C
```

```
C       TKE - TURBULENT KINETIC ENERGY
C
C     SOURCE TERMS
C
550   CONTINUE
      DO 551 J = 2,M2
      DO 551 I = 2,L2
         RHOTED = RHOCON*TED(I,J)
         CON(I,J) = 1.5*GK(I,J) + C2M*RHOTED
         AP(I,J) = -(0.5*GK(I,J) + C2*RHOTED)/(TKE(I,J)+1.E-20)
  551 CONTINUE
C
C       WALL TERMS
C
      DO 553 I = 2,L2
         UP = 0.5*(U(I+1,M2) + U(I,M2))
         UPY = UP/YDIF(M1)
         TAUNW(I) = TAUN(I)*UP
         CON(I,M2) = TAUNW(I)*UPY
         AP(I,M2) = -CD*RHOCON**2*TKE(I,M2)*UPY/(TAUNW(I)+1.E-20)
  553 CONTINUE
      DO 555 J = 1,M1
      DO 555 I = 1,L1
         GAM(I,J) = AMU + AMUT(I,J)/PRTKE
  555 CONTINUE
      DO 556 I = 1,L1
         GAM(I,1) = 0.0
  556 CONTINUE
      DO 557 J = 1,M1
         GAM(L1,J) = 0.0
  557 CONTINUE
      RETURN
C
C       TED - TURBULENT ENERGY DISSIPATION
C
C       SOURCE TERMS
C
560   DO 561 J = 2,M2
      DO 561 I = 2,L2
         TEOK = TED(I,J)/(TKE(I,J)+1.E-20)
         RHOTED = RHOCON*TED(I,J)
         CON(I,J) = (C1*GK(I,J) + C2M*RHOTED)*TEOK
         AP(I,J) = -TC2M*RHOCON*TEOK
  561 CONTINUE
C
C       WALL TERMS
C
      DO 563 I = 2,L2
         CON(I,M2) = CD75*ABS(TKE(I,M2)**1.5)*1.E30
     :               /(CAPPA*YDIF(M1))
         AP(I,M2) = -1.E30
  563 CONTINUE
      DO 565 J = 1,M1
      DO 565 I = 1,L1
         GAM(I,J) = AMU + AMUT(I,J)/PRTED
  565 CONTINUE
      DO 566 I = 1,L1
         GAM(I,1) = 0.0
  566 CONTINUE
      DO 567 J = 1,M1
         GAM(L1,J) = 0.0
  567 CONTINUE
      RETURN
C
599   RETURN
C
```

```
C     ----------
      ENTRY WALL
C     ----------
C
C       AJP = An,   AJM = As,   AIP = Ae,   AIM = Aw
C
      GO TO (700,720,799,799,750,760), NF
C     ------------
C
C       U - VELOCITY
C
700   DO 702 I = 2,L2
      AJP(I,M2) = (TAUN(I-1)*FXM(I) + TAUN(I)*FX(I))*XDIF(I)*Y(M1)
  702 CONTINUE
      RETURN
C
C       V - VELOCITY
C
720   CONTINUE
      RETURN
C
C       TKE - TURBULENT KINETIC ENERGY
C
750   CONTINUE
      RETURN
C
C       TED - TURBULENT ENERGY DISSIPATION
C
760   CONTINUE
      RETURN
799   RETURN
C     ------------
      ENTRY OUTPUT
C     ------------
      IF (LBUG) PRINT *, "ENTER OUTPUT"
C
      IF (ITER.NE.LAST) RETURN
C
C     CREATE RESTART FILE
C
      REWIND 12
      WRITE(12) ITER
      DO 20 K = 1,NFD
       DO 20 J = 1,M1
        DO 20 I = 1,L1
           WRITE(12) F(I,J,K)
   20    CONTINUE
C
      CLOSE(UNIT=12,STATUS='KEEP')
      REFP = P(L1,2)
      DO 22 J = 1,M1
       DO 22 I = 1,L1
          P(I,J) = P(I,J) - REFP
   22    CONTINUE
C
C       RATIO OF VOLUME INFLOW TO VOLUME OUTFLOW
C
      QOUT = 0.0
      DO 23 J = 2,M1
          QOUT = QOUT + (U(L1-1,J-1) + U(L1-1,J))*DY2(J)
   23 CONTINUE
      QINOUT = QIN/(QOUT + 1.0e-20)
C
      LOUT = L1 - 3
      DO 24 J = 1,M1
         UFRIC = SQRT(TAUNW(LOUT)/RHOCON) + 1.0E-20
```

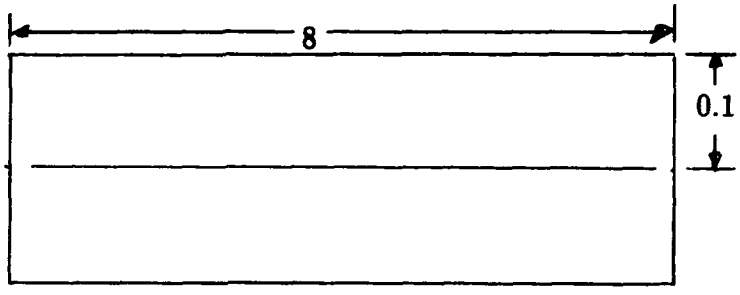
```
          YFRIC = AMU/SQRT(RHOCON*TAUNW(LOUT)) + 1.0E-20
          UOUF(J) = U(LOUT,J)/UFRIC
          YOYF(J) = (Y(M1) - Y(J))/YFRIC
          UOUMAX(J) = U(LOUT,J)/U(LOUT,1)
          YOR(J) = Y(J)/Y(M1)
   24 CONTINUE
C
      NH = M1/2 + 1
      NL = L1/2 + 1
606   FORMAT(7(1X,e9.3))
      WRITE(IO,612) QINOUT
612   FORMAT(/5X,'QIN/QOUT =',F6.2)
      WRITE(IO,614)
614   FORMAT(/5X,'MID-LENGTH U DISTRIBUTION',/)
      WRITE(IO,606) (U(NL,J),J=1,M1)
      WRITE(IO,615)
615   FORMAT(/5X,'AXIAL U DISTRIBUTION',/)
      WRITE(IO,606) (U(I,1),I=2,L1)
      WRITE(IO,616)
616   FORMAT(/5X,'LEFT - U DISTRIBUTION',/)
      WRITE(IO,606) (U(2,J),J=1,M1)
      WRITE(IO,618)
618   FORMAT(/5X,'RIGHT - U DISTRIBUTION',/)
      WRITE(IO,606) (U(L1,J),J=1,M1)
      WRITE(IO,619) REYN
619   FORMAT(//,2X,'REYNOLDS NUMBER = ',E10.4,//,
     :       5X,'(YMAX-Y)/YFRIC',/)
      WRITE(IO,606) (YOYF(J), J=1,M1)
      WRITE(IO,620)
620   FORMAT(//,5X,'U/UFRIC AT I = L1 - 3',/)
      WRITE(IO,606) (UOUF(J), J=1,M1)
      WRITE(IO,621)
621   FORMAT(//,5X,'Y(J)/Y(M1)',/)
      WRITE(IO,636) (YOR(J), J=1,M1)
      WRITE(IO,622)
622   FORMAT(//,5X,'U/UMAX AT I = L1 - 3',/)
      WRITE(IO,606) (UOUMAX(J), J=1,M1)
      WRITE(IO,625)
625   FORMAT(/5X,'X  NODE LOCATIONS',/)
      WRITE(IO,636) (X(I),I=1,L1)
      WRITE(IO,626)
626   FORMAT(/5X,'Y NODE LOCATIONS',/)
      WRITE(IO,636) (Y(I),I=1,M1)
      WRITE(IO,627)
627   FORMAT(/5X,'XU LOCATIONS',/)
      WRITE(IO,636) (XU(I),I=1,L1)
      WRITE(IO,628)
628   FORMAT(/5X,'YV LOCATIONS',/)
      WRITE(IO,636) (YV(I),I=1,M1)
636   FORMAT(8(1X,F8.5))
      WRITE(IO,640) DIST
640   FORMAT(//,5X,'I,   Y+,   TAU/U,   TAU,   DELTA,   DIST =',
     :       e10.4,,/)
      DO 60 I = 1,L1
      WRITE(IO,642) I,PLUN(I),TAUN(I),TAUNW(I),DELTA(I)
642   FORMAT(4X,I2,3(1X,F8.2),1X,E10.4)
   60 CONTINUE
      WRITE(IO,644)
      WRITE(IO,606) (TKE(1,J), J=1,M1)
644   FORMAT(//,5X,'TKE AT I = 1',/)
      WRITE(IO,645)
      WRITE(IO,606) (TED(1,J), J=1,M1)
645   FORMAT(//,5X,'TED AT I = 1',/)
      WRITE(IO,646)
      WRITE(IO,606) (TKE(NL,J), J=1,M1)
646   FORMAT(//,5X,'TKE AT MID-LENGTH',/)
```

```
      WRITE(IO,647)
      WRITE(IO,606) (TED(NL,J), J=1,M1)
647   FORMAT(//,5X,'TED AT MID-LENGTH',/)
      WRITE(IO,648)
      WRITE(IO,606) (TKE(LOUT,J), J=1,M1)
648   FORMAT(//,5X,'TKE AT I = L1 - 3',/)
      WRITE(IO,649)
      WRITE(IO,606) (TED(LOUT,J), J=1,M1)
649   FORMAT(//,5X,'TED AT I = L1 - 3',/)

      WRITE(IO,656)LAST,L1,M1
656   FORMAT(/5X,'NO. ITERATIONS =',
     + I5,2X,'L1,M1 - ',2I3,/)
      WRITE(IO,657)(I,NTIMES(I),I=1,NFP3)
657   FORMAT(/2X,'EQN SWEEPS',/,6(5X,I2,1X,I2))
      IF (ITER.EQ.LAST) LSTOP=.TRUE.
C
C      WRITE DATA TO FILE 17 FOR USE BY PLOT PACKAGE
C
      WRITE(17,680) L1,M1
      WRITE(17,681) (X(I),I=1,L1)
      WRITE(17,681) (Y(I),I=1,M1)
      WRITE(17,681) (XU(I),I=1,L1)
      WRITE(17,681) (YV(I),I=1,M1)
       RHO=RHOCON
       DO 4000 J = 1,M1
       WRITE(17,682) (RHO,I=1,L1)
C      WRITE(IO,606) (CON(I,J),I=LFAB,L3)
 4000  CONTINUE
       DO 4010 J = 1,M1
       WRITE(17,682) (U(I,J),I=1,L1)
 4010  CONTINUE
       DO 4020 J = 1,M1
       WRITE(17,682) (V(I,J),I=1,L1)
 4020  CONTINUE
       DO 4030 J = 1,M1
       WRITE(17,682) (F(I,J,5),I=1,L1)
 4030  CONTINUE
 680   FORMAT(1X,2I5)
 681   FORMAT(7(1X,E10.4))
 682   FORMAT(6(1X,E12.6))
      RETURN
      END
```

Output for turbulent pipe flow, Case 1.

$$U_{in} = U_{max}(1 - R/R_{max})^{1/9}$$



$\rho = 1.0 \qquad \mu = 0.0001$

COMPUTATION FOR AXISYMMETRIC SITUATION
****

QIN/QOUT = 1.00

MID-LENGTH U DISTRIBUTION

0.279E+03 0.279E+03 0.276E+03 0.273E+03 0.269E+03 0.266E+03 0.262E+03
0.258E+03 0.254E+03 0.249E+03 0.245E+03 0.240E+03 0.235E+03 0.230E+03
0.224E+03 0.219E+03 0.213E+03 0.206E+03 0.199E+03 0.192E+03 0.184E+03
0.174E+03 0.162E+03 0.147E+03 0.118E+03 0.000E+00

AXIAL U DISTRIBUTION

0.293E+03 0.289E+03 0.289E+03 0.288E+03 0.288E+03 0.287E+03 0.287E+03
0.286E+03 0.286E+03 0.285E+03 0.285E+03 0.284E+03 0.283E+03 0.283E+03
0.282E+03 0.281E+03 0.281E+03 0.280E+03 0.280E+03 0.279E+03 0.279E+03
0.278E+03 0.278E+03 0.277E+03 0.277E+03 0.277E+03 0.277E+03 0.277E+03
0.276E+03 0.276E+03 0.276E+03 0.276E+03 0.276E+03 0.277E+03 0.277E+03
0.277E+03 0.277E+03 0.277E+03 0.277E+03 0.277E+03 0.277E+03

LEFT - U DISTRIBUTION

0.293E+03 0.290E+03 0.284E+03 0.278E+03 0.272E+03 0.267E+03 0.261E+03
0.256E+03 0.250E+03 0.245E+03 0.240E+03 0.235E+03 0.229E+03 0.224E+03
0.219E+03 0.214E+03 0.208E+03 0.203E+03 0.197E+03 0.191E+03 0.185E+03
0.178E+03 0.170E+03 0.159E+03 0.139E+03 0.000E+00

RIGHT - U DISTRIBUTION

0.277E+03 0.277E+03 0.276E+03 0.273E+03 0.270E+03 0.267E+03 0.262E+03
0.258E+03 0.253E+03 0.248E+03 0.244E+03 0.238E+03 0.233E+03 0.228E+03
0.223E+03 0.217E+03 0.212E+03 0.206E+03 0.200E+03 0.194E+03 0.186E+03
0.178E+03 0.168E+03 0.152E+03 0.128E+03 0.000E+00

REYNOLDS NUMBER = 0.5000E+06

(YMAX-Y)/YFRIC

0.988E+04 0.899E+04 0.741E+04 0.615E+04 0.510E+04 0.423E+04 0.350E+04
0.289E+04 0.239E+04 0.197E+04 0.162E+04 0.133E+04 0.108E+04 0.882E+03
0.714E+03 0.574E+03 0.457E+03 0.360E+03 0.280E+03 0.213E+03 0.157E+03
0.110E+03 0.713E+02 0.391E+02 0.122E+02 0.000E+00

U/UFRIC AT I = L1 - 3

0.281E+02 0.281E+02 0.279E+02 0.277E+02 0.274E+02 0.270E+02 0.266E+02
0.262E+02 0.257E+02 0.252E+02 0.247E+02 0.242E+02 0.237E+02 0.231E+02
0.226E+02 0.220E+02 0.214E+02 0.207E+02 0.200E+02 0.193E+02 0.184E+02
0.174E+02 0.163E+02 0.148E+02 0.118E+02 0.000E+00

Y(J)/Y(M1)

0.00000 0.08989 0.24951 0.37728 0.48365 0.57220 0.64592 0.70729
0.75837 0.80091 0.83631 0.86579 0.89033 0.91075 0.92776 0.94192
0.95370 0.96352 0.97168 0.97848 0.98414 0.98886 0.99278 0.99605
0.99876 1.00000

U/UMAX AT I = L1 - 3

0.100E+01 0.100E+01 0.995E+00 0.986E+00 0.975E+00 0.963E+00 0.948E+00
0.933E+00 0.917E+00 0.900E+00 0.882E+00 0.863E+00 0.844E+00 0.825E+00
0.804E+00 0.783E+00 0.761E+00 0.738E+00 0.713E+00 0.686E+00 0.656E+00
0.622E+00 0.580E+00 0.527E+00 0.422E+00 0.000E+00

YPLUS

0.467E+04 0.425E+04 0.351E+04 0.291E+04 0.241E+04 0.200E+04 0.165E+04
0.137E+04 0.113E+04 0.930E+03 0.765E+03 0.627E+03 0.512E+03 0.417E+03
0.337E+03 0.271E+03 0.216E+03 0.170E+03 0.132E+03 0.101E+03 0.741E+02
0.521E+02 0.337E+02 0.185E+02 0.577E+01 0.000E+00

UPLUS AT I = L1 - 3

0.593E+02 0.593E+02 0.590E+02 0.585E+02 0.579E+02 0.571E+02 0.563E+02
0.554E+02 0.544E+02 0.534E+02 0.523E+02 0.512E+02 0.501E+02 0.489E+02
0.477E+02 0.465E+02 0.452E+02 0.438E+02 0.423E+02 0.407E+02 0.389E+02
0.369E+02 0.344E+02 0.312E+02 0.250E+02 0.000E+00

X NODE LOCATIONS

0.00000 0.10000 0.30000 0.50000 0.70000 0.90000 1.10000 1.30000
1.50000 1.70000 1.90000 2.10000 2.30000 2.50000 2.70000 2.90000
3.10000 3.30000 3.50000 3.70000 3.90000 4.10000 4.30000 4.50000
4.70000 4.90000 5.10000 5.30000 5.50000 5.70000 5.90000 6.10000
6.30000 6.50000 6.70000 6.90000 7.10000 7.30000 7.50000 7.70000
7.90000 8.00000

Y NODE LOCATIONS

0.00000 0.00899 0.02495 0.03773 0.04836 0.05722 0.06459 0.07073
0.07584 0.08009 0.08363 0.08658 0.08903 0.09108 0.09278 0.09419
0.09537 0.09635 0.09717 0.09785 0.09841 0.09889 0.09928 0.09960
0.09988 0.10000

XU LOCATIONS

0.00000 0.00000 0.20000 0.40000 0.60000 0.80000 1.00000 1.20000
1.40000 1.60000 1.80000 2.00000 2.20000 2.40000 2.60000 2.80000
3.00000 3.20000 3.40000 3.60000 3.80000 4.00000 4.20000 4.40000
4.60000 4.80000 5.00000 5.20000 5.40000 5.60000 5.80000 6.00000
6.20000 6.40000 6.60000 6.80000 7.00000 7.20000 7.40000 7.60000
7.80000 8.00000

YV LOCATIONS

0.00000 0.00000 0.01798 0.03192 0.04353 0.05320 0.06124 0.06794
0.07352 0.07816 0.08202 0.08524 0.08792 0.09015 0.09200 0.09355
0.09483 0.09591 0.09680 0.09754 0.09816 0.09867 0.09910 0.09946
0.09975 0.10000

I, Y+, TAU/U, TAU, DELTA, DIST =0.1235E-03 TAUWALL = 0.2182E+02

1	0.00	0.00	0.00	0.0000E+00
2	13.73	0.92	125.21	0.1080E-03
3	12.94	0.88	110.59	0.1146E-03
4	12.43	0.85	101.46	0.1192E-03
5	12.38	0.85	100.70	0.1197E-03
6	12.34	0.85	100.01	0.1201E-03
7	12.31	0.85	99.57	0.1204E-03
8	12.29	0.85	99.29	0.1206E-03
9	12.28	0.85	99.13	0.1207E-03
10	12.28	0.85	99.05	0.1207E-03

11	12.28	0.85	99.02	0.1207E-03
12	12.28	0.85	99.04	0.1207E-03
13	12.28	0.85	99.08	0.1207E-03
14	12.29	0.85	99.15	0.1206E-03
15	12.29	0.85	99.22	0.1206E-03
16	12.29	0.85	99.30	0.1206E-03
17	12.30	0.85	99.38	0.1205E-03
18	12.30	0.85	99.45	0.1205E-03
19	12.31	0.85	99.51	0.1204E-03
20	12.31	0.85	99.56	0.1204E-03
21	12.31	0.85	99.60	0.1204E-03
22	12.32	0.85	99.63	0.1204E-03
23	12.32	0.85	99.64	0.1203E-03
24	12.32	0.85	99.64	0.1204E-03
25	12.31	0.85	99.63	0.1204E-03
26	12.31	0.85	99.59	0.1204E-03
27	12.31	0.85	99.55	0.1204E-03
28	12.31	0.85	99.48	0.1204E-03
29	12.30	0.85	99.41	0.1205E-03
30	12.30	0.85	99.32	0.1205E-03
31	12.29	0.85	99.22	0.1206E-03
32	12.28	0.85	99.11	0.1207E-03
33	12.28	0.84	98.99	0.1207E-03
34	12.27	0.84	98.88	0.1208E-03
35	12.26	0.84	98.77	0.1209E-03
36	12.26	0.84	98.66	0.1209E-03
37	12.25	0.84	98.56	0.1210E-03
38	12.25	0.84	98.56	0.1210E-03
39	12.20	0.84	97.60	0.1215E-03
40	12.60	0.86	104.79	0.1176E-03
41	13.08	0.89	113.43	0.1133E-03
42	0.00	0.00	0.00	0.0000E+00

TKE AT I = 1

0.421E+03 0.421E+03 0.403E+03 0.387E+03 0.371E+03 0.356E+03 0.341E+03
0.327E+03 0.314E+03 0.300E+03 0.288E+03 0.275E+03 0.263E+03 0.251E+03
0.240E+03 0.228E+03 0.217E+03 0.206E+03 0.195E+03 0.183E+03 0.171E+03
0.158E+03 0.144E+03 0.126E+03 0.971E+02 0.000E+00

TED AT I = 1

0.259E+06 0.259E+06 0.243E+06 0.228E+06 0.214E+06 0.201E+06 0.189E+06
0.178E+06 0.167E+06 0.156E+06 0.146E+06 0.137E+06 0.128E+06 0.119E+06
0.111E+06 0.104E+06 0.960E+05 0.887E+05 0.815E+05 0.744E+05 0.672E+05
0.597E+05 0.517E+05 0.423E+05 0.287E+05 0.000E+00

TKE AT MID-LENGTH

0.755E+02 0.755E+02 0.851E+02 0.961E+02 0.106E+03 0.116E+03 0.126E+03
0.136E+03 0.147E+03 0.158E+03 0.170E+03 0.182E+03 0.196E+03 0.209E+03
0.223E+03 0.237E+03 0.251E+03 0.265E+03 0.278E+03 0.292E+03 0.305E+03
0.320E+03 0.339E+03 0.380E+03 0.331E+03 0.000E+00

TED AT MID-LENGTH

0.684E+04 0.684E+04 0.868E+04 0.113E+05 0.146E+05 0.190E+05 0.248E+05
0.327E+05 0.433E+05 0.577E+05 0.770E+05 0.103E+06 0.137E+06 0.183E+06
0.245E+06 0.327E+06 0.439E+06 0.593E+06 0.810E+06 0.113E+07 0.162E+07
0.244E+07 0.404E+07 0.810E+07 0.201E+08 0.000E+00

TKE AT I = L1 - 3

0.579E+02 0.579E+02 0.636E+02 0.724E+02 0.835E+02 0.962E+02 0.110E+03
0.125E+03 0.141E+03 0.156E+03 0.172E+03 0.187E+03 0.202E+03 0.216E+03
0.230E+03 0.244E+03 0.257E+03 0.270E+03 0.282E+03 0.295E+03 0.308E+03
0.323E+03 0.344E+03 0.387E+03 0.325E+03 0.000E+00

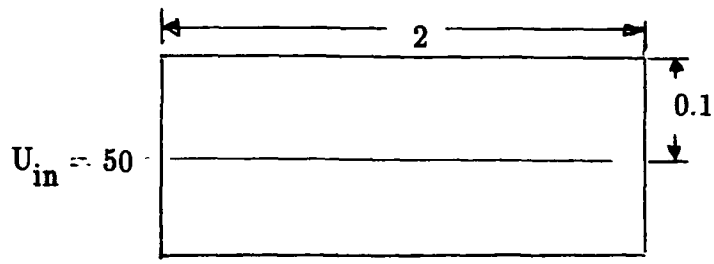
TED AT I = L1 - 3

0.414E+04 0.414E+04 0.526E+04 0.733E+04 0.106E+05 0.154E+05 0.221E+05
0.314E+05 0.438E+05 0.603E+05 0.818E+05 0.110E+06 0.146E+06 0.194E+06
0.256E+06 0.339E+06 0.450E+06 0.603E+06 0.818E+06 0.113E+07 0.163E+07
0.247E+07 0.411E+07 0.815E+07 0.195E+08 0.000E+00

NO. ITERATIONS = 1000 L1,M1 - 42 26

EQN SWEEPS
1 10 2 10 3 10 4 10 5 10 6 10

Output for turbulent pipe flow Case 3.



$\rho = 1.0$ $\mu = 0.0001$

COMPUTATION FOR AXISYMMETRIC SITUATION
****

QIN/QOUT = 1.00

MID-LENGTH U DISTRIBUTION

0.523E+02 0.523E+02 0.523E+02 0.523E+02 0.522E+02 0.521E+02 0.516E+02
0.498E+02 0.443E+02 0.000E+00

AXIAL U DISTRIBUTION

0.500E+02 0.507E+02 0.512E+02 0.518E+02 0.523E+02 0.528E+02 0.532E+02
0.536E+02 0.536E+02

LEFT - U DISTRIBUTION

0.500E+02 0.500E+02 0.500E+02 0.500E+02 0.500E+02 0.500E+02 0.500E+02
0.500E+02 0.500E+02 0.000E+00

RIGHT - U DISTRIBUTION

0.536E+02 0.536E+02 0.536E+02 0.536E+02 0.534E+02 0.530E+02 0.518E+02
0.487E+02 0.425E+02 0.000E+00

REYNOLDS NUMBER = 0.1000E+06

(YMAX-Y)/YFRIC

0.239E+04 0.224E+04 0.194E+04 0.164E+04 0.134E+04 0.104E+04 0.746E+03
0.447E+03 0.149E+03 0.000E+00

U/UFRIC AT I = L1 - 3

0.221E+02 0.221E+02 0.221E+02 0.221E+02 0.221E+02 0.220E+02 0.217E+02
0.207E+02 0.182E+02 0.000E+00

Y(J)/Y(M1)

0.00000 0.06250 0.18750 0.31250 0.43750 0.56250 0.68750 0.81250
0.93750 1.00000

U/UMAX AT I = L1 - 3

0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.998E+00 0.995E+00 0.982E+00
0.938E+00 0.823E+00 0.000E+00

X NODE LOCATIONS

0.00000 0.12500 0.37500 0.62500 0.87500 1.12500 1.37500 1.62500
1.87500 2.00000

Y NODE LOCATIONS

0.00000 0.00625 0.01875 0.03125 0.04375 0.05625 0.06875 0.08125
0.09375 0.10000

XU LOCATIONS

0.00000 0.00000 0.25000 0.50000 0.75000 1.00000 1.25000 1.50000

```
1.75000  2.00000

   YV LOCATIONS

0.00000  0.00000  0.01250  0.02500  0.03750  0.05000  0.06250  0.07500
0.08750  0.10000


   I,    Y+,    TAU/U,    TAU,    DELTA,    DIST =0.6250E-02

    1      0.00      0.00      0.00 0.0000E+00
    2    164.38      0.14      7.09 0.4563E-03
    3    162.34      0.14      6.77 0.4620E-03
    4    158.01      0.14      6.41 0.4746E-03
    5    154.39      0.14      6.12 0.4858E-03
    6    151.37      0.13      5.89 0.4955E-03
    7    148.87      0.13      5.69 0.5038E-03
    8    147.42      0.13      5.59 0.5088E-03
    9    146.98      0.13      5.56 0.5103E-03
   10      0.00      0.00      0.00 0.0000E+00


   TKE AT I = 1

0.125E+02 0.125E+02 0.125E+02 0.125E+02 0.125E+02 0.125E+02 0.125E+02
0.125E+02 0.125E+02 0.000E+00


   TED AT I = 1

0.133E+04 0.133E+04 0.133E+04 0.133E+04 0.133E+04 0.133E+04 0.133E+04
0.133E+04 0.133E+04 0.000E+00


   TKE AT MID-LENGTH

0.352E+01 0.352E+01 0.350E+01 0.350E+01 0.351E+01 0.360E+01 0.411E+01
0.689E+01 0.196E+02 0.000E+00


   TED AT MID-LENGTH

0.148E+03 0.148E+03 0.147E+03 0.147E+03 0.149E+03 0.164E+03 0.259E+03
0.892E+03 0.568E+04 0.000E+00


   TKE AT I = L1 - 3

0.301E+01 0.301E+01 0.299E+01 0.299E+01 0.301E+01 0.315E+01 0.390E+01
0.720E+01 0.189E+02 0.000E+00


   TED AT I = L1 - 3

0.111E+03 0.111E+03 0.110E+03 0.110E+03 0.114E+03 0.133E+03 0.246E+03
0.932E+03 0.541E+04 0.000E+00

   NO. ITERATIONS = 500  L1,M1 -  10 10
```

APPENDIX C
Test Chamber

This and the following appendices through F describe the routines used to model the test chamber. This appendix has a list of terms used in both the flow field model and the agent transport model. The terms associated with the flow field model are given first, those for the agent transport model that differ follow. This list of terms is to be used in conjunction with the list of variables given in Appendix A. The following two figures illustrate the meaning of some of the indices used to define the grid structure of the model.

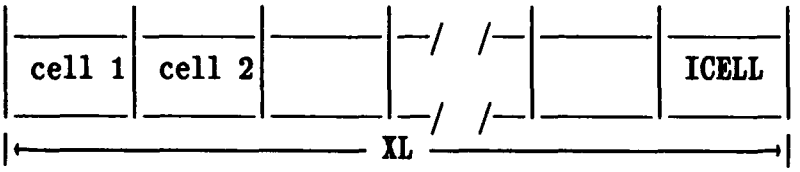


Figure C1. The subdivision of an x-direction region of length XL into ICELL's. A y-direction region is treated in the same way.

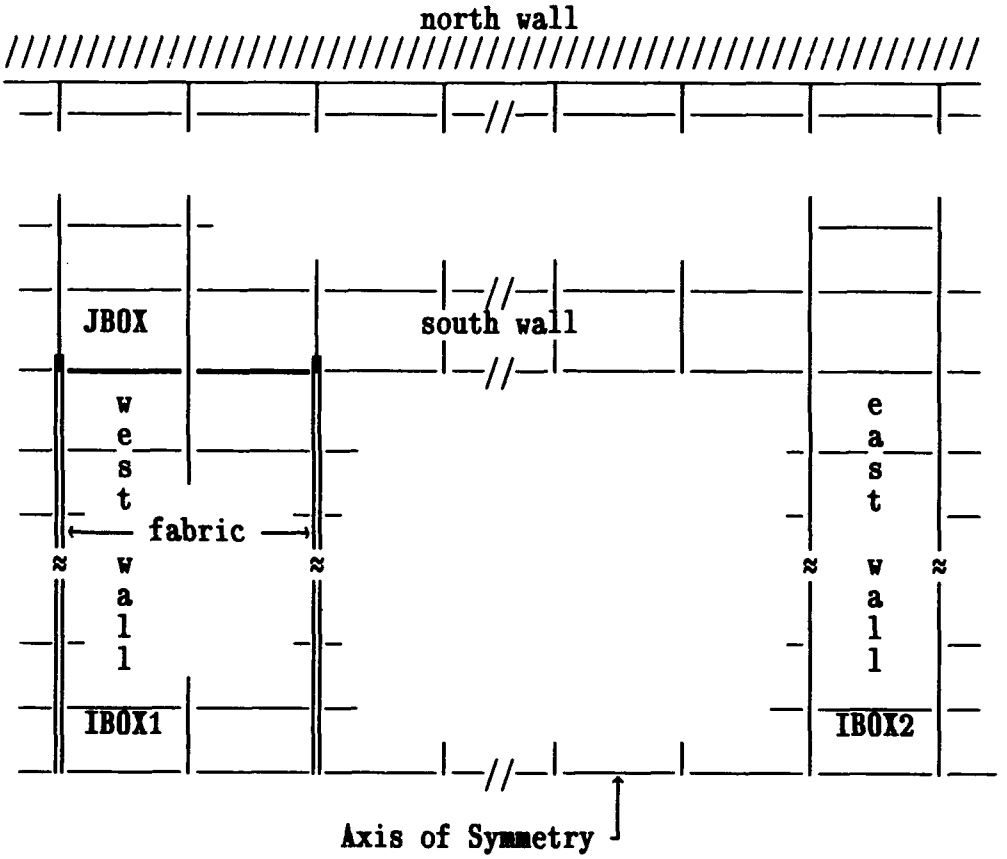


Figure C2. The test chamber showing the cell (control volume) numbering scheme. The north wall is the outer wall of the main tube. Wall functions are used at the north, south, east, and west walls.

```
C
C     VARIABLES USED IN THE MODEL
C
C     ----- PRIMARY UNKNOWNS -----
C
C     U(I,J)          AXIAL VELOCITY
C     V(I,J)          RADIAL VELOCITY
C     P(I,J)          PRESSURE
C     PC(I,J)         PRESSURE CORRECTION
C     TED(I,J)        TURBULENT ENERGY DISSIPATION
C     TKE(I,J)        TURBULENT KINETIC ENERGY
C
C     ----- SECONDARY UNKNOWNS -----
C
C     AMUT(I,J)       TURBULENT VISCOSITY
C     GK(I,J)         2*AMUT*[e]:[e] -- [e] IS THE STRAIN RATE TENSOR
C
C                   ----- INPUT QUANTITIES -----
C
C     NSTEADY         = 1 STEADY STATE PROBLEM
C                     = 0 TRANSIENT PROBLEM
C     DT              TIME STEP -- ENTER IF NSTEADY = 1
C     LAST            MAXIMUM NUMBER OF ITERATIONS OR TIME STEPS
C     NTIMES(NEQ)     NUMBER OF SOLUTION SWEEPS FOR EACH EQUATION
C                     BETWEEN UPDATES
C     RELAX(NEQ)      RELAXATION FACTORS FOR EACH EQUATION
C     MODE            = 1  CARTESIAN COORDINATES
C                     = 2  AXISYMMETRIC COORDINATES
C                     = 3  POLAR COORDINATES
C     IPREF,JPREF     GRID POINT LOCATION FOR THE REFERENCE PRESSURE
C
C       ----- PHYSICAL PROPERTIES AND TURBULENCE CONSTANTS -----
C
C     RHOCON          CONSTANT VALUE OF DENSITY
C     AMU             LAMINAR VISCOSITY
C     C1,C2           CONSTANTS IN THE ENERGY DISSIPATION EQUATION
C     CD              CONSTANT IN THE TURBULENT VISCOSITY FORMULA
C     CAPPA           VON KARMAN'S CONSTANT IN THE WALL LAW
C     ECON            WALL LAW ROUGHNESS CONSTANT
C     PRTKE           PRANDTL NUM IN THE KINETIC ENERGY EQUATION
C     PRTED           PRANDTL NUM IN THE ENERGY DISSIPATION EQUATION
C
C     ALP             ALPHA IN THE ARMOUR-CANNON EQUATION
C     BET             BETA IN THE ARMOUR-CANNON EQUATION
C     ARAT            SURFACE-AREA-TO-VOLUME RATIO FOR THE FABRIC
C     DIA             PORE DIAMETER FOR THE FABRIC
C     EPS             VOID FRACTION FOR THE FABRIC
C     FABTHK          EFFECTIVE THICKNESS OF THE FABRIC
C
C     UINLET          AVERAGE INLET VELOCITY
C     UINJECT         TEST CELL INJECTION VELOCITY
C
C                     ----- MESH INPUT DATA -----
C
C     IREG            NUMBER OF X - DIRECTION MESH REGIONS
C     ICELL(I)        NUMBER OF CELLS IN EACH X - REGION
C     IDCRS(I)        = 1 DECREASING MESH SPACING
C                     = 0 INCREASING MESH SPACING IN EACH REGION
C     JREG            NUMBER OF Y - DIRECTION MESH REGIONS
C     JCELL(J)        NUMBER OF CELLS IN EACH Y - REGION
C     JDCRS(J)        = 1 DECREASING MESH SPACING
C                     = 0 INCREASING MESH SPACING IN EACH REGION
C     XL(I)           LENGTH OF EACH X - REGION
C     ENX(I)          VALUE OF N IN POWER LAW OF THE GRID
C                     (S**(1/N))
C     YL(J)           LENGTH OF EACH Y - REGION
```

```
C     ENY(J)        VALUE OF N IN POWER LAW OF THE GRID
C                   (S**(1/N))
C
C                   ----- TEST CELL LOCATION -----
C
C     IBOX1               FIRST I-DIR CELL (FABRIC)
C     IBOX2               LAST I-DIR CELL AT BACK WALL
C     JBOX                J-DIR CELL AT TOP-WALL
C
C     GRID POINT RANGES FOR IMPERVIOUS BOX WALLS
C     AND THE RANGE FOR THE INJECTION VELOCITY
C
C     LB1 = IBOX1 + 1
C     LB2 = IBOX2 + 1
C     MB1 = 1
C     MB2 = JBOX + 1
C     MU1 = MB1
C     MU2 = MB2 - 1
C
C     RANGE FOR GRID POINTS NEXT TO N,S,E,W WALLS RESPECTIVELY
C
C     LNW1 = 1
C     LNW2 = L1
C     MNW  = M2
C     LSW1 = LB1
C     LSW2 = LB2
C     MSW  = MB2 + 1
C     LEW  = LB1 - 1
C     MEW1 = 1
C     MEW2 = MB2
C     LWW  = LB2 + 1
C     MWW1 = 1
C     MWW2 = MEW2
C
C     GRID POINT RANGES FOR FABRIC AND
C     THE WIDTH OF THE FABRIC CELLS
C
C     LF1 = LB1
C     LF2 = LB1 + 1
C     MF1 = 1
C     MF2 = MB2 - 1
C     WF1 = XCV(LF1)
C     WF2 = XCV(LF2)
C
C     OUTER RADIUS AND DISTANCES FROM WALLS TO
C     ADJACENT GRID POINT
C
C     RMAX  = Y(M1)
C     DISTN = YDIF(M1)
C     DISTS = Y(MSW) - YV(MSW)
C     DISTE = XU(LEW+1) - X(LEW)
C     DISTW = X(LWW) - XU(LWW)
C
C           VARIABLES FOR THE TRANSPORT OF AGENT MODEL
C           THAT DIFFER FROM THE MODEL FOR THE VELOCITY
C
C     ----- PRIMARY UNKNOWN -----
C
C     AGNT(I,J)       CONCENTRATION OF AGENT
C
C                   -----  INPUT QUANTITIES -----
C     -----  PHYSICAL PROPERTIES AND TURBULENT CONSTANTS -----
C
C     ETA     DIFFUSION COEFFICIENT FOR FLUID
C     ETAF    DIFFUSION COEFFICIENT FOR FABRIC
C     PRCON   TURBULENT SCHMIDT NUMBER
```

APPENDIX D
User Routine for Velocity

The structure of the user routines for the chamber model follows that given in the chart in Appendix A. It differs slightly from that used in Appendix B. There is no BLOCK DATA section. Data are either defined or read in the subroutine USER. Output is controlled by the subroutine PUTOUT in Appendix E. The model given here is for the case of radial injection of uncontaminated gas. The range of nodes in the axial direction along which the injection takes place is defined by LV1 and LV2 defined on page 102. The injection values are enforced after ENTRY BOUND on pages 103 and 104 and the corresponding coefficients in the finite difference equation are given the appropriate values following ENTRY GAMSOR on page 106.

Following the listing is output for the case with an inlet velocity of $U_{inlet} = 50in/s$ and and injection velocity of $V_{inj} = 20in/s$ as discussed in the main body of the text.

```
C
C.....................................................................
C
C            ------   T E S T   C H A M B E R   ------
C
C                    T U R B U L E N T   F L O W
C
C.....................................................................
C
      SUBROUTINE USER
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
      PARAMETER (ID=151,JD=151,NFD=7,NFP3=7,MIJ=151,NREG=10)
C
      LOGICAL  LSOLVE,LBLK,LSTOP
C
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     : AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     : X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     : Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     : YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     : R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
C
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     : FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
C
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     : IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
C
      COMMON/LOGI/LSOLVE(NFP3),LBLK(NFP3)
C
      COMMON/VARI/TIME,DT,RELAX(NFP3),RHOCON
C
      COMMON/CNTL/LSTOP
C
      COMMON/SORC/SMAX,SSUM
C
      COMMON/COEF/FLOW,DIFF,ACOF
C
      COMMON/INOUT/IN,IO
C
      COMMON/STRESS/TOUN(ID),TOUS(ID),TOVE(ID),TOVW(ID),
     :              TAUN(ID),TAUS(ID),TAUE(ID),TAUW(ID),
     :              TOUNI(ID),TOVEI(ID),TOVWI(ID),
     :              TAUNI(ID),TAUEI(ID),TAUWI(ID),AMUT(ID,JD)
C
      COMMON/PROP/ AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED,
     :             ALP,BET,ARAT,DIA,EPS,FABTHK,UINLET, UINJECT,
     :             CD5,CD25,CD75,C2M,TC2M,CDR2
C
      COMMON/BOX/ WF1,WF2,SCFAC,AA,BB,DISTN,DISTS,DISTE,DISTW,
     :            DISTNI,DISTEI,DISTWI
C
      COMMON/IBOX/ LB1,LB2,MB2,LNW1,LNW2,LSW1,LSW2,MNW,MSW,
     :             MEW1,MEW2,MWW1,MWW2,LEW,LWW,LF1,LF2,MF1,MF2,
     :             MNWI,LWWI,LEWI
C
      DIMENSION U(ID,JD),V(ID,JD),PC(ID,JD),P(ID,JD),TKE(ID,JD),
     :          TED(ID,JD),GK(ID,JD)
C
      DIMENSION TURBE(ID)
C
      DIMENSION ICELL(NREG),IDCRS(NREG),JCELL(NREG),JDCRS(NREG),
     :          XL(NREG),YL(NREG),ENX(NREG),ENY(NREG)
C
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
```

```
     :             (F(1,1,3),PC(1,1)),(F(1,1,4),P(1,1)),
     :             (F(1,1,5),TKE(1,1)),(F(1,1,6),TED(1,1))
C
C         F(I,J,1) = U
C         F(I,J,2) = V
C         F(I,J,3) = PC
C         F(I,J,4) = P
C         F(I,J,5) = TKE
C         F(I,J,6) = TED
C
C...PROBLEM DATA
C
      DATA IN,IO /15,16/
      DATA NFMIN,NFMAX /5,6/
      DATA NU,NV,NPC,NP,NKE,NED /1,2,3,4,5,6/
C
      DATA LSOLVE(1) /.true./
      DATA LSOLVE(2) /.true./
      DATA LSOLVE(3) /.true./
      DATA LSOLVE(4) /.true./
      DATA LSOLVE(5) /.true./
      DATA LSOLVE(6) /.true./
      DATA LSOLVE(7) /.FALSE./
C
      DATA LBLK(1)  /.FALSE./
      DATA LBLK(2)  /.FALSE./
      DATA LBLK(3)  /.FALSE./
      DATA LBLK(4)  /.FALSE./
      DATA LBLK(5)  /.FALSE./
      DATA LBLK(6)  /.FALSE./
      DATA LBLK(7)  /.FALSE./
C
      DATA LSTOP /.FALSE./
      DATA TINY,BIG,SMALLEST,BIGGEST /1.E-30, 1.E+30, 1.E-70, 1.E+70/
C
C     -----------
      ENTRY INPUT
C     -----------
C
      READ(IN,*) NSTEADY
       DT = BIG
       IF (NSTEADY .NE. 1) READ(5,*) DT
      READ(IN,*) LAST, (NTIMES(NEQ), NEQ=1,NFP3)
      READ(IN,*) (RELAX(NEQ), NEQ=1,NFP3)
      READ(IN,*) MODE, IPREF, JPREF
      READ(IN,*) RHOCON,AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED
      READ(IN,*) ALP,BET,ARAT,DIA,EPS,FABTHK
      READ(IN,*) UINLET, UINJECT
C
      WRITE(IO,*) 'NSTEADY'
      WRITE(IO,6100) NSTEADY
      WRITE(IO,*) 'DT'
      WRITE(IO,6102) DT
      WRITE(IO,*) 'LAST, (NTIMES(NEQ), NEQ=1,NFP3)'
      WRITE(IO,6100) LAST, (NTIMES(NEQ), NEQ=1,NFP3)
      WRITE(IO,*) 'MODE, IPREF, JPREF'
      WRITE(IO,6100)  MODE, IPREF, JPREF
      WRITE(IO,*) 'RHOCON,AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED'
      WRITE(IO,6102) RHOCON,AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED
      WRITE(IO,*) 'ALP,BET,ARAT,DIA,EPS,FABTHK'
      WRITE(IO,6102) ALP,BET,ARAT,DIA,EPS,FABTHK
      WRITE(IO,*) 'UINLET, UINJECT'
      WRITE(IO,6102)  UINLET, UINJECT
C
6100  FORMAT(15I5,/)
6102  FORMAT(7(1X,E10.4),/)
```

```
C
C       TURBULENCE CONSTANTS
C
      CD5  = CD**.5
      CD25 = CD**.25
      CD75 = CD**.75
      C2M  = C2 - 1.0
      TC2M = 2.*C2 - 1.0
      CDR2 = CD*RHOCON**2
      COK  = BIG*CD75/CAPPA
C
C
C     MESH GENERATION DATA
C
      read(in,*) ireg, (icell(i), idcrs(i), i=1,ireg)
      read(in,*) jreg, (jcell(j), jdcrs(j), j=1,jreg)
      read(in,*) (xl(i), enx(i), i=1,ireg)
      read(in,*) (yl(j), eny(j), j=1,jreg)
C
C     TEST CELL
C
      READ(IN,*) IBOX1,IBOX2,JBOX
C
C     ----------
      ENTRY GRID
C     ----------
C
C
C        L1       NUMBER OF U VELOCITY POINTS
C        M1       NUMBER OF V VELOCITY POINTS
C
C
      call meshgen(ireg,icell,idcrs,xl,enx,xu,l1)
      call meshgen(jreg,jcell,jdcrs,yl,eny,yv,m1)
C
      RETURN
C
C     -----------
      ENTRY START
C     -----------
C
C      OPEN RESTART FILE
C
       OPEN(UNIT=12,FILE='RESTART',FORM='UNFORMATTED')
C
C
      DO 10  K = 1,NFD
       DO 10  J = 1,M1
        DO 10  I = 1,Li
         F(I,J,K) = 0.0
  10  CONTINUE
C
C     GRID POINT RANGES FOR IMPERVIOUS BOX WALLS
C     AND THE RANGE FOR THE INJECTION VELOCITY
C
      LB1 = IBOX1 + 1
      LB2 = IBOX2 + 1
      MB1 = 1
      MB2 = JBOX + 1
C
C     RANGE FOR GRID POINTS NEXT TO N,S,E,W WALLS RESPECTIVELY
C
      LNW1 = 1
      LNW2 = L1
      MNW  = M2
      LSW1 = LB1
      LSW2 = LB2
```

```
      MSW  = MB2 + 1
      LEW  = LB1 - 1
      MEW1 = 1
      MEW2 = MB2
      LWW  = LB2 + 1
      MWW1 = 1
      MWW2 = MEW2
      MNWI = MB2 - 1
      LWWI = LB1 + 2
      LEWI = LB2 - 1
C
C     GRID POINT RANGES FOR FABRIC AND
C     THE WIDTH OF THE FABRIC CELLS
C
      LF1 = LB1
      LF2 = LB1 + 1
      MF1 = 1
      MF2 = MB2 - 1
      WF1 = XCV(LF1)
      WF2 = XCV(LF2)
C
C     GRID POINT RANGES FOR INJECTION VELOCITY
C
      LV1 = IBOX1 + 5
      LV2 = IBOX2 - 2
C
C     SCALED COEFFICIENTS FOR THE ARMOUR-CANNON EQUATION
C
      SCFAC = FABTHK/(WF1 + WF2)
      AA = SCFAC*ALP*ARAT**2/EPS**2
      BB = SCFAC*BET*RHOCON/(DIA*EPS**2)
C
C     OUTER RADIUS AND DISTANCES FROM WALLS TO
C     ADJACENT GRID POINT
C
      RMAX  = Y(M1)
      DISTN = YDIF(M1)
      DISTS = Y(MSW) - YV(MSW)
      DISTE = XU(LEW+1) - X(LEW)
      DISTW = X(LWW) - XU(LWW)
      DISTNI = Y(MNWI) - YV(MNWI)
      DISTEI = XU(LEWI+1) - X(LEWI)
      DISTWI = X(LWWI) - XU(LWWI)
C
C     WALL FACTORS
C
      FCTRN = COK/DISTN
      FCTRS = COK/DISTS
      FCTRE = COK/DISTE
      FCTRW = COK/DISTW
      FCTRNI = COK/DISTNI
      FCTREI = COK/DISTEI
      FCTRWI = COK/DISTWI
C
      WRITE(IO,*) '  AA, BB, SCFAC'
      WRITE(IO,6102) AA, BB, SCFAC
      WRITE(IO,*) '  RMAX, DISTN, DISTS, DISTE, DISTW'
      WRITE(IO,6102) RMAX, DISTN, DISTS, DISTE, DISTW
      WRITE(IO,*) '  DISTNI, DISTEI, DISTWI'
      WRITE(IO,6102) DISTNI, DISTEI, DISTWI
      WRITE(IO,*) '  FCTRN, FCTRS, FCTRE, FCTRW'
      WRITE(IO,6102) FCTRN, FCTRS, FCTRE, FCTRW
      WRITE(IO,*) '  FCTRNI, FCTREI, FCTRWI'
      WRITE(IO,6102) FCTRNI, FCTREI, FCTRWI
      WRITE(IO,*) '  IBOX1, IBOX2, JBOX'
      WRITE(IO,6100) IBOX1, IBOX2, JBOX
```

```
      WRITE(IO,*) '  LB1, LB2, MB1, MB2'
      WRITE(IO,6100) LB1, LB2, MB1, MB2
      WRITE(IO,*) '  LNW1, LNW2, MNW, LSW1, LSW2, MSW'
      WRITE(IO,6100) LNW1, LNW2, MNW, LSW1, LSW2, MSW
      WRITE(IO,*) '  LEW, MEW1, MEW2, LWW, MWW1, MWW2'
      WRITE(IO,6100) LEW, MEW1, MEW2, LWW, MWW1, MWW2
      WRITE(IO,*) '  LEWI, LWWI, MNWI'
      WRITE(IO,6100) LEWI, LWWI, MNWI
      WRITE(IO,*) '  LF1, LF2, MF1, MF2'
      WRITE(IO,6100) LF1, LF2, MF1, MF2
      WRITE(IO,*) '  WF1, WF2'
      WRITE(IO,6102) WF1, WF2
      WRITE(IO,*) '  LV1, LV2'
      WRITE(IO,6100) LV1, LV2
C
C           ----- INITIAL CONDITIONS -----
C
      DO 11 J = 1,M2
       DO 11 I = 1,L1
        U(I,J) = UINLET
        TKE(I,J) = 0.005*U(I,J)**2
        TED(I,J) = CD*TKE(I,J)**1.5/(0.03*RMAX)
   11 CONTINUE
       DO 12 J = 1,MB2
       DO 12 I = LB1,LB2
        U(I,J) = 0.0
        TKE(I,J) = 0.0
        TED(I,J) = 0.0
   12 CONTINUE
      RETURN
C
C     -----------
      ENTRY DENSE
C     -----------
C
      DO 200 J = 1,M1
       DO 200 I = 1,L1
        DENOM = ABS(TED(I,J)) + TINY
        AMUT(I,J) = CD*RHOCON*TKE(I,J)**2/DENOM
  200 CONTINUE
      DO 202 J = 1,MB2
       DO 202 I = LB1,LB2
        AMUT(I,J) = 0.0
  202 CONTINUE
      RETURN
C
C     -----------
      ENTRY BOUND
C     -----------
C
C        AXIS AND OUTER WALL
C
      DO 300 I = 2,L1
         U(I,1)    = U(I,2)
         U(I,M1)   = 0.0
         V(I,2)    = 0.0
         V(I,M1)   = 0.0
         TKE(I,1)  = TKE(I,2)
         TKE(I,M1) = 0.0
         TED(I,1)  = TED(I,2)
         TED(I,M1) = 0.0
         TURBE(I)  = ABS(TKE(I,MNW))
  300    CONTINUE
C
     DO 301 I = LV1,LV2
        V(I,2) = UINJECT
```

```
          V(I,3) = UINJECT
  301     CONTINUE
C
      CALL WALLSTRESS(ECON,CAPPA,CD25,AMU,RHOCON,DISTN,TURBE,TOUN,2,L1)
C
C         INLET AND OUTLET
C
      DO 302 J=1,M2
          U(2,J)     = UINLET
          U(L1,J)    = U(L2,J)
          V(1,J)     = 0.0
          V(L1,J)    = 0.0
          TKE(1,J)   = 0.005*U(2,J)**2
          TKE(L1,J)  = TKE(L2,J)
          TED(1,J)   = CD*TKE(1,J)**1.5/(0.03*RMAX)
          TED(L1,J)  = TED(L2,J)
  302     CONTINUE
C
C         OUTER AND BACK WALLS OF CHAMBER
C
      DO 304 I = LB1,LB2
          U(I,MB2)    = 0.0
          V(I,MB2)    = 0.0
          V(I,MB2+1)  = 0.0
          TKE(I,MB2)  = 0.0
          TED(I,MB2)  = 0.0
          TURBE(I)    = ABS(TKE(I,MSW))
  304     CONTINUE
C
      CALL WALLSTRESS(ECON,CAPPA,CD25,AMU,RHOCON,DISTS,TURBE,TOUS,
     :                LB1,LB2)
C
      DO 306 J = 1,MB2
          U(LB2,J)    = 0.0
          U(LB2+1,J)  = 0.0
          V(LB2,J)    = 0.0
          TKE(LB2,J)  = 0.0
          TED(LB2,J)  = 0.0
          TURBE(J)    = ABS(TKE(LWW,J))
  306     CONTINUE
C
      CALL WALLSTRESS(ECON,CAPPA,CD25,AMU,RHOCON,DISTW,TURBE,TOVW,
     :                1,MB2)
C
C
C         FABRIC
C
      DO 308 J = 1,MB2
          V(LF1,J) = 0.0
          V(LF2,J) = 0.0
          TURBE(J)  = ABS(TKE(LEW,J))
  308     CONTINUE
C
      CALL WALLSTRESS(ECON,CAPPA,CD25,AMU,RHOCON,DISTE,TURBE,TOVE,
     :                1,MB2)
C
C         --------------
          ENTRY GRADIENT
C         --------------
C         COMPUTE GENK=GK(I,J) AT MAIN GRID POINTS FOR USE
C         IN TURBULENT ENERGY AND DISSIPATION EQUATIONS
C
        DO 404 J = 2,M2
        DO 402 I = 2,L2
           UPX = (U(I+1,J) - U(I,J))/XCV(I)
           VPY = (V(I,J+1) - V(I,J))/YCV(J)
```

```
            VOR = 0.5*(V(I,J+1) + V(I,J))/Y(J)
            UNSIDE = 0.5*((U(I,J) + U(I+1,J))*FYM(J+1)
     :             + (U(I,J+1) + U(I+1,J+1))*FY(J+1))
            USSIDE = 0.5*((U(I,J-1) + U(I+1,J-1))*FYM(J)
     :             + (U(I,J) + U(I+1,J))*FY(J))
            VESIDE = 0.5*((V(I,J) + V(I,J+1))*FXM(I+1)
     :             + (V(I+1,J) + V(I+1,J+1))*FX(I+1))
            VWSIDE = 0.5*((V(I-1,J) + V(I-1,J+1))*FXM(I)
     :             + (V(I,J) + V(I,J+1))*FX(I))
            UPY = (UNSIDE - USSIDE)/YCV(J)
            VPX = (VESIDE - VWSIDE)/XCV(I)
C
            GK(I,J) = AMUT(I,J)*(2.*(UPX*UPX + VPX*VPX + VOR*VOR)
     :              + (UPY + VPX)**2)
  402    CONTINUE
  404    CONTINUE
         RETURN
C
C        ------------
         ENTRY GAMSOR
C        ------------
C
C
C            S = SC + SP*U  SO, AP = SP  &  CON = SC
C
         GO TO (500,550,600,799,650,700),NF
C
C                  -------- U - VELOCITY --------
C
500      CONTINUE
         DO 502 J = 1,M1
         DO 501 I = 1,L1
            GAM(I,J) = AMU + AMUT(I,J)
  501    CONTINUE
            GAM(L1,J) = 0.0
  502    CONTINUE
         DO 504 I = 2,L1
            GAM(I,1) = 0.0
  504    CONTINUE
C                              BOX TOP
         DO 505 I = LB1,LB2
            AP(I,MB2) = -BIG
  505    CONTINUE
C                              BOX BACK
         DO 506 J =  1,MB2
            AP(LB2,J) = -BIG
            AP(LB2+1,J) = -BIG
  506    CONTINUE
C                              FABRIC
         DO 508 J = 1,MF2
            I = LF1
            AP(I,J) = -0.5*XCV(I)*AA*AMU
            I = LF2
            AP(I,J) = -XDIF(I)*AA*AMU
            I = LF2 + 1
            AP(I,J) = -0.5*XCV(I-1)*AA*AMU
  508    CONTINUE
C                              INJECTOR
         RETURN
C
C                  -------- V - VELOCITY --------
C
550      CONTINUE
         DO 552 J = 1,M1
         DO 552 I = 1,L1
            GAM(I,J) = AMU + AMUT(I,J)
            GAM(1,J) = 0.0
```

```
  552    CONTINUE
C                                  BOX TOP
       DO 555 I = LB1,LB2
          AP(I,MB2)       = -BIG
          AP(I,MB2+1)     = -BIG
  555    CONTINUE
C                                  BOX FRONT & BACK
       DO 556 J = 2,MB2
          AP(LF1,J) = -BIG
          AP(LF2,J) = -BIG
          AP(LB2,J) = -BIG
  556    CONTINUE
C                                 INJECTOR
       DO 559 J = 2,3
       DO 559 I = LV1,LV2
          CON(I,J) =  BIG*UINJECT
          AP(I,J)  = -BIG
  559    CONTINUE
       RETURN
C
C            -------- PRESSURE CORRECTION --------
C
600    CONTINUE
       RETURN
C
C          -------- TKE - TURBULENT KINETIC ENERGEY --------
C
C                          SOURCE TERMS
C
650    CONTINUE
       DO 651 J = 2,M2
       DO 651 I = 2,L2
          RHOTED = RHOCON*TED(I,J)
          CON(I,J) = 1.5*GK(I,J) + C2M*RHOTED
          AP(I,J) = -(0.5*GK(I,J) + C2*RHOTED)/(TKE(I,J)+TINY)
  651    CONTINUE
C
C                          WALL TERMS
C
       DO 653 I = 2,L2
          UP = ABS(0.5*(U(I+1,MNW) + U(I,MNW)))
          UPY = UP/DISTN
          TAUN(I) = ABS(TOUN(I)*UP)
          CON(I,MNW) = TAUN(I)*UPY
          AP(I,MNW) = -CDR2*TKE(I,MNW)*UPY/(TAUN(I)+TINY)
  653    CONTINUE
C
       DO 655 I = LSW1,LSW2
          UP = ABS(0.5*(U(I+1,MSW) + U(I,MSW)))
          UPY = UP/DISTS
          TAUS(I) = ABS(TOUS(I)*UP)
          CON(I,MSW) = TAUS(I)*UPY
          AP(I,MSW) = -CDR2*TKE(I,MSW)*UPY/(TAUS(I)+TINY)
  655    CONTINUE

       DO 657 J = MEW1,MEW2
          VP = ABS(0.5*(V(LEW,J+1) + V(LEW,J)))
          VPX = VP/DISTE
          TAUE(J) = ABS(TOVE(J)*VP)
          CON(LEW,J) = TAUE(J)*VPX
          AP(LEW,J) = -CDR2*TKE(LEW,J)*VPX/(TAUE(J)+TINY)
  657    CONTINUE
C
       DO 659 J = MWW1,MWW2
          VP = ABS(0.5*(V(LWW,J+1) + V(LWW,J)))
          VPX = VP/DISTW
```

```
          TAUW(J) = ABS(TOVW(J)*VP)
          CON(LWW,J) = TAUW(J)*VPX
          AP(LWW,J) = -CDR2*TKE(LWW,J)*VPX/(TAUW(J)+TINY)
  659     CONTINUE
C
        DO 665 J = 1,M1
        DO 665 I = 1,L1
           GAM(I,J) = AMU + AMUT(I,J)/PRTKE
  665      CONTINUE
        DO 666 I = 1,L1
           GAM(I,1) = 0.0
  666     CONTINUE
C
        DO 667 J = 1,M1
           GAM(L1,J) = 0.0
  667     CONTINUE
C                        BOX
        DO 675 J = 1,MB2
         DO 675 I = LB1,LB2
           AP(I,J) = -BIG
  675     CONTINUE
        RETURN
C
C         -------- TED - TURBULENT ENERGY DISSIPATION --------
C
C         SOURCE TERMS
C
  700   DO 701 J = 2,M2
        DO 701 I = 2,L2
           DENOM = ABS(TKE(I,J)+TINY)
           TEOK = ABS(TED(I,J))/DENOM
           RHOTED = RHOCON*ABS(TED(I,J))
           CON(I,J) = (C1*GK(I,J) + C2M*RHOTED)*TEOK
           AP(I,J) = -TC2M*RHOCON*TEOK
  701 CONTINUE
C
C         WALL TERMS
C
        DO 703 I = 2,L2
           ATK = ABS(TKE(I,MNW))
           CON(I,MNW) = FCTRN*ATK*SQRT(ATK)
           AP(I,MNW) = -BIG
  703 CONTINUE
        DO 705 I = LSW1,LSW2
           ATK = ABS(TKE(I,MSW))
           CON(I,MSW) = FCTRS*ATK*SQRT(ATK)
           AP(I,MSW) = -BIG
  705 CONTINUE
        DO 707 J = MEW1,MEW2
           ATK = ABS(TKE(LEW,J))
           CON(LEW,J) = FCTRE*ATK*SQRT(ATK)
           AP(LEW,J) = -BIG
  707 CONTINUE
        DO 709 J = MWW1,MWW2
           ATK = ABS(TKE(LWW,J))
           CON(LWW,J) = FCTRW*ATK*SQRT(ATK)
           AP(LWW,J) = -BIG
  709 CONTINUE
C
        DO 725 J = 1,M1
        DO 725 I = 1,L1
           GAM(I,J) = AMU + AMUT(I,J)/PRTED
  725 CONTINUE
        DO 726 I = 1,L1
           GAM(I,1) = 0.0
  726 CONTINUE
```

```
      DO 727 J = 1,M1
         GAM(L1,J) = 0.0
  727 CONTINUE
C                          BOX
      DO 735 J = 1,MB2
       DO 735 I = LB1,LB2
         AP(I,J) = -BIG
  735    CONTINUE
      RETURN
C
799   RETURN
C
C     ----------
      ENTRY WALL
C     ----------
C
C       AJP = An,  AJM = As,  AIP = Ae,  AIM = Aw
C
      GO TO (800,820,899,899,850,860), NF
C     ------------
C
C       U - VELOCITY
C
800   CONTINUE
      DO 802 I = 2,L2
      AJP(I,MNW) = (TOUN(I-1)*FXM(I) + TOUN(I)*FX(I))*XDIF(I)*YV(M1)
  802 CONTINUE
      DO 803 I = LB1,LB2
      AJM(I,MSW) = (TOUS(I-1)*FXM(I) + TOUS(I)*FX(I))*XDIF(I)*YV(MSW)
  803 CONTINUE
      RETURN
C
C       V - VELOCITY
C
820   CONTINUE
      DO 823 J = MEW1,MEW2
      AIP(LEW,J) = (TOVE(J-1)*FYM(J) + TOVE(J)*FY(J))*YDIF(J)*YV(J)
  823 CONTINUE
      DO 824 J = MWW1,MWW2
      AIM(LWW,J) = (TOVW(J-1)*FYM(J) + TOVW(J)*FY(J))*YDIF(J)*YV(J)
  824 CONTINUE
      RETURN
C
C       TKE - TURBULENT KINETIC ENERGY
C
850   CONTINUE
      RETURN
C
C       TED - TURBULENT ENERGY DISSIPATION
C
860   CONTINUE
      RETURN
899   RETURN
C     ------------
      ENTRY OUTPUT
C     ------------
C
      IF (ITER.NE.LAST) RETURN
C
      CALL PUTOUT
C
      RETURN
      END
c
      subroutine meshgen(nreg,ncell,ndcrs,el,en,x,nxs)
      implicit real*8 (a-h,o-z)
```

```
      implicit integer(i-n)
      dimension ncell(1),ndcrs(1),el(1),en(1),x(1)
c
      x(1) = 0.0
      x(2) = 0.0
      mo = 2
      mf = ncell(1) + mo
      nxs = mo
c
      do 1 n = 1,nreg
        mf = ncell(n) + mo
        x(mf) = x(mo) + el(n)
        pwr = 1.0/en(n)
        nx = ncell(n)
        nsgn = ndcrs(n)
        nxs = nxs + nx
        call coord(pwr,nx,nsgn,mo,mf,x)
        mo = mf
        mf = ncell(n+1) + mo
    1 continue
      return
      end
c
      subroutine coord(pwr,nx,nsgn,mo,mf,x)
      implicit real*8 (a-h,o-z)
      implicit integer(i-n)
      dimension x(1),si(100)
c
      dsi = 1.0/nx
      si(1) = 0.0
      si(nx+1) = 1.0
      do 1 i = 2,nx
        si(i) = si(i-1) + dsi
    1 continue
      do 2 i = 1,nx
        if (nsgn .ge. 0) then
          si(i+1) = si(i+1)**pwr
        else
          si(i+1) = 1.0 - (1.0 - si(i+1))**pwr
        endif
    2 continue
      xo = x(mo)
      xf = x(mf)
      nx1 = nx + 1
      do 3 i = 1,nx1
        ix = mo+i-1
        x(ix) = (xf - xo)*si(i) + xo
    3 continue
      return
      end
c
      subroutine wallstress(e,cappa,cd25,amu,rho,dist,tke,tauovel,k1,k2)
      implicit real*8 (a-h,o-z)
      implicit integer(i-n)
      dimension tke(1),tauovel(1)
c
c     computes wall stress divided by the velocity
c
      do 1 i = k1,k2
         rk = rho*sqrt(tke(i))
         yplus = rk*cd25*dist/amu
         if (yplus .gt. 11.5) then
            tauovel(i) = cappa*rk*cd25/dlog(e*yplus)
         else
            tauovel(i) = amu/dist
         endif
```

```
1 continue
  return
  end
```

```
NSTEADY
    1
DT
 0.1000E+31
LAST, (NTIMES(NEQ), NEQ=1,NFP3)
 1000    1    1    1    1    1    1    0
MODE, IPREF, JPREF
    2  120    2
RHOCON,AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED
 0.1062E-06 0.2750E-08 0.1430E+01 0.1920E+01 0.9000E-01 0.4000E+00 0.9000E+01

 0.1000E+01 0.1300E+01
ALP,BET,ARAT,DIA,EPS,FABTHK
 0.8610E+01 0.4200E+00 0.1490E+03 0.2770E-02 0.4300E+00 0.2680E-01
UINLET, UINJECT
 0.5000E+02 0.2000E+02
              COMPUTATION FOR AXISYMMETRIC SITUATION
             ****************************************


  AA, BB, SCFAC
 0.9235E+05 0.7780E-05 0.8933E-01
  RMAX, DISTN, DISTS, DISTE, DISTW
 0.4000E+01 0.3207E-01 0.3207E-01 0.1206E+00 0.1206E+00
  DISTNI, DISTEI, DISTWI
 0.4069E-01 0.7500E-01 0.7500E-01
  FCTRN, FCTRS, FCTRE, FCTRW
 0.1281E+32 0.1281E+32 0.3406E+31 0.3406E+31
  FCTRNI, FCTREI, FCTRWI
 0.1010E+32 0.5477E+31 0.5477E+31
  IBOX1, IBOX2, JBOX
   51   70   10
  LB1, LB2, MB1, MB2
   52   71    1   11
  LNW1, LNW2, MNW, LSW1, LSW2, MSW
    1  122   31   52   71   12
  LEW, MEW1, MEW2, LWW, MWW1, MWW2
   51    1   11   72    1   11
  LEWI, LWWI, MNWI
   70   54   10
  LF1, LF2, MF1, MF2
   52   53    1   10
  WF1, WF2
 0.1500E+00 0.1500E+00
  LV1, LV2
   56   68
  WROTE F(I,J,K) TO RESTART  L1, M1, NFD
  122   32    7
  WROTE AMUT(I,J) TO RESTART L1, M1
  122   32

     QIN/QOUT =  0.99

     EAST WALL -  U

 0.188E+02 0.188E+02 0.152E+02 0.144E+02 0.146E+02 0.154E+02 0.170E+02
 0.194E+02 0.230E+02 0.283E+02 0.361E+02 0.447E+02 0.507E+02 0.546E+02
 0.573E+02 0.589E+02 0.599E+02 0.605E+02 0.607E+02 0.608E+02 0.609E+02
 0.605E+02 0.597E+02 0.585E+02 0.563E+02 0.536E+02 0.505E+02 0.471E+02
 0.429E+02 0.366E+02 0.259E+02 0.000E+00

     WEST WALL - U

-.876E+01 -.876E+01 -.629E+01 -.469E+01 -.323E+01 -.172E+01 -.920E-01
 0.168E+01 0.361E+01 0.644E+01 0.142E+02 0.329E+02 0.447E+02 0.503E+02
 0.548E+02 0.586E+02 0.617E+02 0.640E+02 0.655E+02 0.660E+02 0.657E+02
```

0.652E+02 0.641E+02 0.627E+02 0.605E+02 0.580E+02 0.550E+02 0.513E+02
0.461E+02 0.379E+02 0.254E+02 0.000E+00

OUTLET - U

0.444E+02 0.444E+02 0.460E+02 0.470E+02 0.481E+02 0.491E+02 0.500E+02
0.509E+02 0.518E+02 0.526E+02 0.534E+02 0.542E+02 0.548E+02 0.556E+02
0.563E+02 0.571E+02 0.579E+02 0.588E+02 0.596E+02 0.603E+02 0.605E+02
0.571E+02 0.514E+02 0.484E+02 0.458E+02 0.432E+02 0.405E+02 0.376E+02
0.339E+02 0.285E+02 0.208E+02 0.000E+00

U AT I = LB2

0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.302E+02 0.429E+02 0.487E+02
0.537E+02 0.580E+02 0.614E+02 0.639E+02 0.654E+02 0.660E+02 0.657E+02
0.652E+02 0.642E+02 0.628E+02 0.607E+02 0.582E+02 0.553E+02 0.518E+02
0.468E+02 0.388E+02 0.262E+02 0.000E+00

U FOR I=LEW,LWW+1 J=1,5,10

0.188E+02 0.162E+02 0.162E+02 0.162E+02 0.423E+02 0.836E+02 0.710E+02
0.583E+02 0.457E+02 0.330E+02 0.204E+02 0.774E+01 -.491E+01 -.176E+02
-.302E+02 -.429E+02 -.555E+02 -.682E+02 -.808E+02 -.358E+02 0.000E+00
0.000E+00 -.876E+01
0.146E+02 0.762E+00 0.763E+00 0.764E+00 0.140E+01 0.257E+00 0.514E+00
0.176E+01 0.757E+01 0.811E+01 0.651E+01 0.303E+01 -.200E+01 -.568E+01
-.741E+01 -.704E+01 0.436E+00 0.144E+01 0.150E+01 0.587E+00 0.000E+00
0.000E+00 -.323E+01
0.283E+02 -.971E+01 -.971E+01 -.971E+01 -.184E+02 -.257E+02 -.312E+02
-.346E+02 -.350E+02 -.315E+02 -.238E+02 -.119E+02 0.893E+01 0.209E+02
0.289E+02 0.322E+02 0.312E+02 0.271E+02 0.205E+02 0.117E+02 0.000E+00
0.000E+00 0.644E+01

V FOR I=LEW,LWW J=3,5,10

0.263E+01 0.000E+00 0.000E+00 -.414E+02 -.653E+02 0.200E+02 0.200E+02
0.200E+02 0.200E+02 0.200E+02 0.200E+02 0.200E+02 0.200E+02 0.200E+02
0.200E+02 0.200E+02 0.200E+02 0.200E+02 -.711E+02 -.566E+02 0.000E+00
0.862E+01
0.143E+02 0.000E+00 0.000E+00 -.268E+02 -.242E+02 -.168E+02 -.846E+01
0.703E+01 0.173E+02 0.242E+02 0.290E+02 0.325E+02 0.296E+02 0.246E+02
0.175E+02 0.692E+01 -.106E+02 -.200E+02 -.278E+02 -.328E+02 0.000E+00
0.112E+02
0.447E+02 0.000E+00 0.000E+00 -.486E+01 -.408E+01 -.311E+01 -.188E+01
-.246E+00 0.196E+01 0.434E+01 0.660E+01 0.117E+02 0.671E+01 0.442E+01
0.185E+01 -.518E+00 -.230E+01 -.369E+01 -.496E+01 -.654E+01 0.000E+00
0.678E+01

AXIAL PRESSURE

0.706E-04 0.687E-04 0.660E-04 0.635E-04 0.615E-04 0.596E-04 0.581E-04
0.568E-04 0.557E-04 0.547E-04 0.538E-04 0.530E-04 0.523E-04 0.516E-04
0.510E-04 0.504E-04 0.499E-04 0.493E-04 0.489E-04 0.484E-04 0.480E-04
0.476E-04 0.472E-04 0.468E-04 0.465E-04 0.462E-04 0.459E-04 0.457E-04
0.455E-04 0.455E-04 0.455E-04 0.456E-04 0.458E-04 0.462E-04 0.468E-04
0.477E-04 0.489E-04 0.505E-04 0.527E-04 0.556E-04 0.594E-04 0.644E-04
0.707E-04 0.788E-04 0.891E-04 0.102E-03 0.117E-03 0.134E-03 0.153E-03
0.171E-03 0.184E-03 0.129E-03 0.366E-04 -.254E-04 -.236E-03 -.649E-03
-.545E-03 -.458E-03 -.388E-03 -.335E-03 -.299E-03 -.280E-03 -.294E-03
-.326E-03 -.375E-03 -.441E-03 -.524E-03 -.624E-03 -.185E-03 0.363E-04
0.159E-04 -.322E-04 -.394E-04 -.425E-04 -.439E-04 -.430E-04 -.396E-04
-.340E-04 -.268E-04 -.188E-04 -.107E-04 -.336E-05 0.286E-05 0.775E-05
0.113E-04 0.137E-04 0.150E-04 0.155E-04 0.155E-04 0.151E-04 0.144E-04
0.136E-04 0.128E-04 0.119E-04 0.111E-04 0.104E-04 0.971E-05 0.909E-05
0.853E-05 0.803E-05 0.758E-05 0.718E-05 0.681E-05 0.647E-05 0.615E-05

0.585E-05 0.555E-05 0.525E-05 0.496E-05 0.466E-05 0.434E-05 0.402E-05
0.367E-05 0.330E-05 0.290E-05 0.247E-05 0.200E-05 0.148E-05 0.870E-06
0.000E+00 0.821E-06 0.140E-05

PRESSURE AT J = 5

0.706E-04 0.687E-04 0.661E-04 0.637E-04 0.616E-04 0.598E-04 0.582E-04
0.569E-04 0.558E-04 0.548E-04 0.539E-04 0.531E-04 0.524E-04 0.517E-04
0.511E-04 0.505E-04 0.499E-04 0.494E-04 0.489E-04 0.485E-04 0.480E-04
0.476E-04 0.472E-04 0.468E-04 0.465E-04 0.462E-04 0.459E-04 0.457E-04
0.455E-04 0.454E-04 0.453E-04 0.453E-04 0.455E-04 0.458E-04 0.462E-04
0.469E-04 0.479E-04 0.492E-04 0.510E-04 0.533E-04 0.563E-04 0.602E-04
0.651E-04 0.715E-04 0.795E-04 0.896E-04 0.102E-03 0.118E-03 0.137E-03
0.159E-03 0.183E-03 0.194E-03 0.189E-03 0.185E-03 0.174E-03 0.163E-03
0.154E-03 0.150E-03 0.164E-03 0.188E-03 0.210E-03 0.222E-03 0.215E-03
0.195E-03 0.173E-03 0.161E-03 0.165E-03 0.172E-03 0.180E-03 0.186E-03
0.861E-04 -.430E-04 -.460E-04 -.478E-04 -.486E-04 -.474E-04 -.439E-04
-.379E-04 -.301E-04 -.216E-04 -.131E-04 -.554E-05 0.739E-06 0.559E-05
0.909E-05 0.114E-04 0.127E-04 0.134E-04 0.135E-04 0.132E-04 0.127E-04
0.121E-04 0.114E-04 0.108E-04 0.101E-04 0.950E-05 0.894E-05 0.842E-05
0.796E-05 0.753E-05 0.715E-05 0.680E-05 0.648E-05 0.618E-05 0.589E-05
0.562E-05 0.534E-05 0.507E-05 0.479E-05 0.450E-05 0.420E-05 0.389E-05
0.355E-05 0.319E-05 0.280E-05 0.238E-05 0.191E-05 0.140E-05 0.831E-06
0.124E-06 0.516E-06 0.793E-06

X NODE LOCATIONS

0.00000 1.69706 4.09706 5.33939 6.33350 7.18885 7.95166 8.64691
9.28999 9.89117 10.45773 10.99506 11.50727 11.99760 12.46863 12.92247
13.36090 13.78537 14.19714 14.59730 14.98676 15.36636 15.73679 16.09870
16.45264 16.79913 17.13860 17.47149 17.79814 18.11891 18.43409 18.74397
19.04881 19.34885 19.64430 19.93537 20.22226 20.50513 20.78415 21.05947
21.33124 21.59959 21.86464 22.12652 22.38534 22.64120 22.89419 23.14443
23.39198 23.63694 23.87939 24.07500 24.22500 24.37500 24.52500 24.67500
24.82500 24.97500 25.12500 25.27500 25.42500 25.57500 25.72500 25.87500
26.02500 26.17500 26.32500 26.47500 26.62500 26.77500 26.92500 27.12061
27.36306 27.60802 27.85557 28.10580 28.35880 28.61466 28.87348 29.13536
29.40041 29.66876 29.94053 30.21585 30.49487 30.77774 31.06463 31.35570
31.65115 31.95119 32.25603 32.56591 32.88109 33.20186 33.52851 33.86140
34.20087 34.54736 34.90130 35.26321 35.63364 36.01324 36.40270 36.80286
37.21463 37.63910 38.07753 38.53137 39.00240 39.49273 40.00494 40.54227
41.10883 41.71001 42.35309 43.04834 43.81115 44.66650 45.66061 46.90294
49.30294 51.00000

Y NODE LOCATIONS

0.00000 0.23717 0.57258 0.74620 0.88513 1.00467 1.11128 1.20844
1.29832 1.38233 1.46151 1.53207 1.59806 1.66807 1.74296 1.82394
1.91277 2.01239 2.12816 2.27285 2.55236 2.94764 3.22715 3.37184
3.48761 3.58723 3.67606 3.75704 3.83193 3.90194 3.96793 4.00000

XU LOCATIONS

0.00000 0.00000 3.39411 4.80000 5.87878 6.78823 7.58947 8.31384
8.97998 9.60000 10.18234 10.73313 11.25700 11.75755 12.23765 12.69961
13.14534 13.57645 13.99428 14.40000 14.79459 15.17893 15.55378 15.91980
16.27759 16.62769 16.97056 17.30665 17.63633 17.95996 18.27786 18.59032
18.89762 19.20000 19.49769 19.79091 20.07984 20.36468 20.64558 20.92271
21.19623 21.46625 21.73292 21.99636 22.25668 22.51400 22.76840 23.01999
23.26886 23.51510 23.75879 24.00000 24.15000 24.30000 24.45000 24.60000
24.75000 24.90000 25.05000 25.20000 25.35000 25.50000 25.65000 25.80000
25.95000 26.10000 26.25000 26.40000 26.55000 26.70000 26.85000 27.00000
27.24121 27.48490 27.73114 27.98001 28.23160 28.48600 28.74332 29.00364
29.26708 29.53375 29.80377 30.07728 30.35442 30.63532 30.92016 31.20909
31.50231 31.80000 32.10238 32.40968 32.72214 33.04004 33.36367 33.69335
34.02944 34.37231 34.72241 35.08020 35.44622 35.82107 36.20541 36.60000

37.00572 37.42355 37.85466 38.30039 38.76235 39.24245 39.74300 40.26687
40.81766 41.40000 42.02002 42.68616 43.41053 44.21177 45.12122 46.20000
47.60589 51.00000

YV LOCATIONS

0.00000 0.00000 0.47434 0.67082 0.82158 0.94868 1.06066 1.16190
1.25499 1.34164 1.42302 1.50000 1.56415 1.63197 1.70417 1.78175
1.86612 1.95943 2.06535 2.19098 2.35472 2.75000 3.14528 3.30902
3.43465 3.54057 3.63388 3.71825 3.79583 3.86803 3.93585 4.00000

EAST WALL - TKE

0.203E+02 0.203E+02 0.251E+02 0.287E+02 0.328E+02 0.376E+02 0.432E+02
0.506E+02 0.606E+02 0.748E+02 0.971E+02 0.168E+03 0.194E+03 0.194E+03
0.169E+03 0.129E+03 0.858E+02 0.502E+02 0.268E+02 0.144E+02 0.795E+01
0.605E+01 0.635E+01 0.886E+01 0.141E+02 0.204E+02 0.260E+02 0.309E+02
0.354E+02 0.410E+02 0.671E+02 0.000E+00

EAST WALL - TED

0.312E+03 0.312E+03 0.429E+03 0.524E+03 0.640E+03 0.785E+03 0.968E+03
0.122E+04 0.161E+04 0.220E+04 0.326E+04 0.555E+04 0.571E+04 0.508E+04
0.391E+04 0.258E+04 0.146E+04 0.686E+03 0.265E+03 0.819E+02 0.180E+02
0.979E+01 0.152E+02 0.375E+02 0.913E+02 0.175E+03 0.283E+03 0.444E+03
0.769E+03 0.174E+04 0.705E+04 0.000E+00

WEST WALL - TKE

0.323E+02 0.323E+02 0.285E+02 0.253E+02 0.220E+02 0.187E+02 0.154E+02
0.124E+02 0.992E+01 0.809E+01 0.181E+02 0.819E+02 0.106E+03 0.976E+02
0.897E+02 0.795E+02 0.673E+02 0.541E+02 0.400E+02 0.247E+02 0.101E+02
0.667E+01 0.754E+01 0.107E+02 0.157E+02 0.208E+02 0.255E+02 0.305E+02
0.368E+02 0.450E+02 0.665E+02 0.000E+00

WEST WALL - TED

0.624E+03 0.624E+03 0.519E+03 0.433E+03 0.350E+03 0.275E+03 0.206E+03
0.149E+03 0.106E+03 0.784E+02 0.262E+03 0.634E+04 0.394E+04 0.251E+04
0.183E+04 0.136E+04 0.989E+03 0.686E+03 0.433E+03 0.216E+03 0.494E+02
0.150E+02 0.215E+02 0.497E+02 0.105E+03 0.185E+03 0.305E+03 0.526E+03
0.100E+04 0.218E+04 0.695E+04 0.000E+00

OUTLET - TKE

0.304E+02 0.304E+02 0.320E+02 0.326E+02 0.329E+02 0.328E+02 0.324E+02
0.317E+02 0.308E+02 0.297E+02 0.285E+02 0.271E+02 0.257E+02 0.241E+02
0.222E+02 0.200E+02 0.175E+02 0.146E+02 0.115E+02 0.864E+01 0.699E+01
0.172E+02 0.288E+02 0.325E+02 0.348E+02 0.362E+02 0.368E+02 0.368E+02
0.362E+02 0.342E+02 0.531E+02 0.000E+00

OUTLET - TED

0.795E+02 0.795E+02 0.912E+02 0.976E+02 0.102E+03 0.104E+03 0.105E+03
0.105E+03 0.103E+03 0.996E+02 0.955E+02 0.908E+02 0.856E+02 0.793E+02
0.717E+02 0.629E+02 0.528E+02 0.417E+02 0.301E+02 0.200E+02 0.167E+02
0.672E+02 0.156E+03 0.200E+03 0.245E+03 0.297E+03 0.368E+03 0.479E+03
0.693E+03 0.128E+04 0.496E+04 0.000E+00

TKE FOR I=LEW,LWW,2 J=1,5,10

```
 0.203E+02 0.266E-68 0.912-138 0.272-206 0.363-275 0.000E+00 0.865-314
 0.325-244 0.273-175 0.988-107 0.000E+00
 0.328E+02 0.104E-70 0.563-143 0.512-216 0.342-287 0.000E+00 0.000E+00
 0.420-265 0.296-192 0.392-110 0.000E+00
 0.748E+02 0.245E-70 0.156E-70 0.122E-70 0.112E-70 0.113E-70 0.117E-70
 0.122E-70 0.127E-70 0.133E-70 0.000E+00

     TED FOR I=LEW,LWW,2  J=1,5,10

 0.312E+03 0.357E-32 0.409E-67 0.468-102 0.140-136 0.830-171 0.417-205
 0.172-239 0.557-274 0.130-308 0.000E+00 0.000E+00 0.160-312 0.129-277
 0.601-243 0.198-208 0.504-174 0.104-139 0.183-105 0.720E-71 0.000E+00
 0.624E+03
 0.640E+03 0.368E-33 0.201E-69 0.110-105 0.109-141 0.258-178 0.991-215
 0.124-250 0.662-286 0.375-321 0.000E+00 0.000E+00 0.000E+00 0.335-299
 0.638-264 0.128-228 0.450-191 0.292-150 0.595-109 0.389E-71 0.000E+00
 0.350E+03
 0.220E+04 0.533E-68 0.358E-68 0.248E-68 0.185E-68 0.149E-68 0.129E-68
 0.118E-68 0.114E-68 0.113E-68 0.114E-68 0.117E-68 0.121E-68 0.124E-68
 0.129E-68 0.133E-68 0.137E-68 0.142E-68 0.146E-68 0.151E-68 0.000E+00
 0.784E+02

     AMUT FOR I=LEW,LWW,2  J=1,5,10

 0.127E-07 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.159E-07
 0.161E-07 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.131E-07
 0.243E-07 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
 0.798E-08
TAUN
 0.372E-05 0.284E-05 0.236E-05 0.208E-05 0.191E-05 0.181E-05 0.174E-05
 0.170E-05 0.167E-05 0.165E-05 0.163E-05 0.162E-05 0.161E-05 0.160E-05
 0.160E-05 0.159E-05 0.159E-05 0.158E-05 0.158E-05 0.157E-05 0.157E-05
 0.157E-05 0.156E-05 0.156E-05 0.156E-05 0.156E-05 0.156E-05 0.155E-05
 0.155E-05 0.155E-05 0.156E-05 0.156E-05 0.156E-05 0.157E-05 0.158E-05
 0.159E-05 0.161E-05 0.162E-05 0.165E-05 0.168E-05 0.171E-05 0.175E-05
 0.180E-05 0.185E-05 0.191E-05 0.198E-05 0.204E-05 0.212E-05 0.219E-05
 0.225E-05 0.230E-05 0.234E-05 0.236E-05 0.239E-05 0.240E-05 0.242E-05
 0.243E-05 0.243E-05 0.243E-05 0.243E-05 0.242E-05 0.241E-05 0.239E-05
 0.238E-05 0.236E-05 0.234E-05 0.231E-05 0.229E-05 0.226E-05 0.224E-05
 0.220E-05 0.215E-05 0.210E-05 0.204E-05 0.199E-05 0.193E-05 0.187E-05
 0.181E-05 0.176E-05 0.171E-05 0.166E-05 0.161E-05 0.158E-05 0.154E-05
 0.152E-05 0.149E-05 0.148E-05 0.147E-05 0.146E-05 0.146E-05 0.146E-05
 0.146E-05 0.146E-05 0.147E-05 0.147E-05 0.148E-05 0.148E-05 0.149E-05
 0.149E-05 0.150E-05 0.150E-05 0.151E-05 0.151E-05 0.151E-05 0.152E-05
 0.152E-05 0.152E-05 0.152E-05 0.152E-05 0.152E-05 0.152E-05 0.152E-05
 0.152E-05 0.152E-05 0.152E-05 0.152E-05 0.152E-05 0.153E-05 0.166E-05
 0.178E-05
TAUS
 0.581E-05 0.403E-05 0.319E-05 0.269E-05 0.236E-05 0.216E-05 0.205E-05
 0.199E-05 0.199E-05 0.201E-05 0.206E-05 0.211E-05 0.217E-05 0.223E-05
 0.230E-05 0.236E-05 0.243E-05 0.249E-05 0.256E-05 0.262E-05
TAUE
 0.000E+00 0.298E-07 0.145E-06 0.304E-06 0.458E-06 0.627E-06 0.826E-06
 0.108E-05 0.142E-05 0.189E-05 0.249E-05
TAUW
 0.000E+00 0.117E-06 0.246E-06 0.267E-06 0.262E-06 0.253E-06 0.237E-06
 0.211E-06 0.175E-06 0.126E-06 0.449E-07
```

```
YPLUSN
 0.579E+01
YPLUSS
 0.563E+01
YPLUSW
 0.120E+02
YPLUSE
 0.146E+02

     NO. ITERATIONS = 1000  L1,M1 - 122 32


  EQN SWEEPS
      1  1       2  1       3  1       4  1       5  1       6  1
      7  0
```

APPENDIX E
User Routine for Output Control

The routine used to control printed output, generation of a file for restart information used to initiate the model for agent transport, and generation of a file for plotting data used by the plotting routines is given here.

```
      SUBROUTINE PUTOUT
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
      PARAMETER (ID=151,JD=151,NFD=7,NFP3=7,MIJ=151)
C
      LOGICAL LSTOP
C
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     : AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     : X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     : Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     : YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     : R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
C
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     : FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
C
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     : IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
C
      COMMON/VARI/TIME,DT,RELAX(NFP3),RHOCON
C
      COMMON/CNTL/LSTOP
C
      COMMON/COEF/FLOW,DIFF,ACOF
C
      COMMON/INOUT/IN,IO
C
      COMMON/STRESS/TOUN(ID),TOUS(ID),TOVE(ID),TOVW(ID),
     :              TAUN(ID),TAUS(ID),TAUE(ID),TAUW(ID),
     :              TOUNI(ID),TOVEI(ID),TOVWI(ID),
     :              TAUNI(ID),TAUEI(ID),TAUWI(ID),AMUT(ID,JD)
C
      COMMON/PROP/ AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED,
     :             ALP,BET,ARAT,DIA,EPS,FABTHK,UINLET, UINJECT,
     :             CD5,CD25,CD75,C2M,TC2M,CDR2
C
      COMMON/BOX/ WF1,WF2,SCFAC,AA,BB,DISTN,DISTS,DISTE,DISTW,
     :            DISTNI,DISTEI,DISTWI
C
      COMMON/IBOX/ LB1,LB2,MB2,LNW1,LNW2,LSW1,LSW2,MNW,MSW,
     :             MEW1,MEW2,MWW1,MWW2,LEW,LWW,LF1,LF2,MF1,MF2,
     :             MNWI,LWWI,LEWI
C
      DIMENSION U(ID,JD),V(ID,JD),PC(ID,JD),P(ID,JD),TKE(ID,JD),
     :          TED(ID,JD),GK(ID,JD)
C
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
     :            (F(1,1,3),PC(1,1)),(F(1,1,4),P(1,1)),
     :            (F(1,1,5),TKE(1,1)),(F(1,1,6),TED(1,1))
C
C         F(I,J,1) = U
C         F(I,J,2) = V
C         F(I,J,3) = PC
C         F(I,J,4) = P
C         F(I,J,5) = TKE
C         F(I,J,6) = TED
C
C
      DATA TINY,BIG /1.E-30, 1.E+30/
C
C     CREATE RESTART FILE
C
      REWIND 12
      WRITE(12) ITER
      DO 10 K = 1,NFD
```

```
       DO 10 J = 1,M1
        DO 10 I = 1,L1
           WRITE(12) F(I,J,K)
   10    CONTINUE
      WRITE(IO,*) '  WROTE F(I,J,K) TO RESTART  L1, M1, NFD '
      WRITE(IO,6100) L1,M1,NFD
       DO 11 J = 1,M1
        DO 11 I = 1,L1
           WRITE(12) AMUT(I,J)
   11    CONTINUE
      WRITE(IO,*) '  WROTE AMUT(I,J) TO RESTART L1, M1 '
      WRITE(IO,6100) L1,M1
6100  FORMAT(10I5)
C
      CLOSE(UNIT=12,STATUS='KEEP')
C
C      CONSTRUCT BOUNDARY PRESSURES BY EXTRAPOLATION
C
      DO 21 J = 2,M2
         P(1,J)  = (P(2,J)*XCVS(3) - P(3,J)*XDIF(2))/XDIF(3)
         P(L1,J) = (P(L2,J)*XCVS(L2) - P(L3,J)*XDIF(L1))/XDIF(L2)
   21    CONTINUE
      DO 22 I = 2,L2
         P(I,1)  = (P(I,2)*YCVS(3) - P(I,3)*YDIF(2))/YDIF(3)
         P(I,M1) = (P(I,M2)*YCVS(M2) - P(I,M3)*YDIF(M1))/YDIF(M2)
   22    CONTINUE
         P(1,1)   = P(2,1) + P(1,2) - P(2,2)
         P(L1,1)  = P(L2,1) + P(L1,2) - P(L2,2)
         P(1,M1)  = P(2,M1) + P(1,M2) - P(2,M2)
         P(L1,M1) = P(L2,M1) + P(L1,M2) - P(L2,M2)
C
      PREF = P(IPREF,JPREF)
C
      DO 23 J = 1,M1
      DO 23 I = 1,L1
         P(I,J) = P(I,J) - PREF
         IF (P(I,J) .LT. -1.E-1) P(I,J) = 0.0
   23    CONTINUE
C
C
C        RATIO OF VOLUME INFLOW TO VOLUME OUTFLOW
C
      QIN = 0.0
      QOUT = 0.0
      DO 25 J = 2,M1
         QIN  = QIN + U(2,J)*ARX(J)
         QOUT = QOUT + U(L2,J)*ARX(J)
   25 CONTINUE
C     DO 26 J = MU1,MU2
C        QIN = QIN + UNIJECT*ARX(J)
C  26    CONTINUE
      QINOUT = QIN/(QOUT + TINY)
C
606   FORMAT(7(1X,e9.3))
      WRITE(IO,612) QINOUT
612   FORMAT(/5X,'QIN/QOUT =',F6.2)
      WRITE(IO,614)
614   FORMAT(/5X,'EAST WALL -  U ',/)
      WRITE(IO,606) (U(LEW,J),J=1,M1)
      WRITE(IO,616)
616   FORMAT(/5X,'WEST WALL - U ',/)
      WRITE(IO,606) (U(LWW+1,J),J=1,M1)
      WRITE(IO,618)
618   FORMAT(/5X,'OUTLET - U ',/)
      WRITE(IO,606) (U(L1,J),J=1,M1)
      WRITE(IO,620)
```

```
620   FORMAT(/,5X,'U AT I = LB2',/)
      WRITE(IO,606) (U(LB2,J), J=1,M1)
      WRITE(IO,622)
622   FORMAT(/,5X, 'U FOR I=LEW,LWW+1   J=1,5,10',/)
      LWP = LWW + 1
         WRITE(IO,606) (U(I,1), I=LEW,LWP)
         WRITE(IO,606) (U(I,5), I=LEW,LWP)
         WRITE(IO,606) (U(I,10), I=LEW,LWP)
      WRITE(IO,624)
624   FORMAT(/,5X, 'V FOR I=LEW,LWW   J=3,5,10',/)
         WRITE(IO,606) (V(I,3), I=LEW,LWW)
         WRITE(IO,606) (V(I,5), I=LEW,LWW)
         WRITE(IO,606) (V(I,10), I=LEW,LWW)
      WRITE(IO,625)
625   FORMAT(/,5X, 'AXIAL PRESSURE',/)
         WRITE(IO,606) (P(I,2), I=1,L1)
      WRITE(IO,626)
626   FORMAT(/,5X, 'PRESSURE AT J = 5',/)
         WRITE(IO,606) (P(I,5), I=1,L1)
      WRITE(IO,627)
627   FORMAT(/5X,'X  NODE LOCATIONS',/)
      WRITE(IO,636) (X(I),I=1,L1)
      WRITE(IO,628)
628   FORMAT(/5X,'Y NODE LOCATIONS',/)
      WRITE(IO,636) (Y(I),I=1,M1)
      WRITE(IO,629)
629   FORMAT(/5X,'XU LOCATIONS',/)
      WRITE(IO,636) (XU(I),I=1,L1)
      WRITE(IO,630)
630   FORMAT(/5X,'YV LOCATIONS',/)
      WRITE(IO,636) (YV(I),I=1,M1)
636   FORMAT(8(1X,F8.5))
      WRITE(IO,644)
      WRITE(IO,606) (TKE(LEW,J), J=1,M1)
644   FORMAT(//,5X,'EAST WALL - TKE ',/)
      WRITE(IO,645)
      WRITE(IO,606) (TED(LEW,J), J=1,M1)
645   FORMAT(//,5X,'EAST WALL - TED',/)
      WRITE(IO,646)
      WRITE(IO,606) (TKE(LWW,J), J=1,M1)
646   FORMAT(//,5X,'WEST WALL - TKE ',/)
      WRITE(IO,647)
      WRITE(IO,606) (TED(LWW,J), J=1,M1)
647   FORMAT(//,5X,'WEST WALL - TED',/)
      WRITE(IO,648)
      WRITE(IO,606) (TKE(L2,J), J=1,M1)
648   FORMAT(//,5X,'OUTLET - TKE ',/)
      WRITE(IO,649)
      WRITE(IO,606) (TED(L2,J), J=1,M1)
649   FORMAT(//,5X,'OUTLET - TED ',/)
C
      WRITE(IO,676)
676   FORMAT(/,5X,'TKE FOR I=LEW,LWW,2  J=1,5,10',/)
         WRITE(IO,606) (TKE(I,1), I=LEW,LWW,2)
         WRITE(IO,606) (TKE(I,5), I=LEW,LWW,2)
         WRITE(IO,606) (TKE(I,10), I=LEW,LWW,2)
      WRITE(IO,678)
678   FORMAT(/,5X,'TED FOR I=LEW,LWW,2  J=1,5,10',/)
         WRITE(IO,606) (TED(I,1), I=LEW,LWW)
         WRITE(IO,606) (TED(I,5), I=LEW,LWW)
         WRITE(IO,606) (TED(I,10), I=LEW,LWW)
C
      WRITE(IO,680)
680   FORMAT(/,5X,'AMUT FOR I=LEW,LWW,2  J=1,5,10',/)
         WRITE(IO,606) (AMUT(I,1), I=LEW,LWW)
         WRITE(IO,606) (AMUT(I,5), I=LEW,LWW)
```

```
          WRITE(IO,606) (AMUT(I,10), I=LEW,LWW)
C
          WRITE(IO,*) 'TAUN'
          WRITE(IO,606) (TAUN(I),I=2,L2)
C
          WRITE(IO,*) 'TAUS'
          WRITE(IO,606) (TAUS(I),I=LSW1,LSW2)
C
          WRITE(IO,*) 'TAUE'
          WRITE(IO,606) (TAUE(J),J=MEW1,MEW2)
C
          WRITE(IO,*) 'TAUW'
          WRITE(IO,606) (TAUW(J),J=MWW1,MWW2)
C
       TURB = TKE(LSW1+10,MNW)
       CALL YYPLUS(CD25,RHOCON,AMU,DISTN,TURB,YPLUS)
            WRITE(IO,*) 'YPLUSN'
            WRITE(IO,606) YPLUS
       TURB = TKE(LSW1+10,MSW)
       CALL YYPLUS(CD25,RHOCON,AMU,DISTS,TURB,YPLUS)
            WRITE(IO,*) 'YPLUSS'
            WRITE(IO,606) YPLUS
C
       TURB = TKE(LWW,5)
       CALL YYPLUS(CD25,RHOCON,AMU,DISTW,TURB,YPLUS)
            WRITE(IO,*) 'YPLUSW'
            WRITE(IO,606) YPLUS
       TURB = TKE(LEW,5)
       CALL YYPLUS(CD25,RHOCON,AMU,DISTE,TURB,YPLUS)
            WRITE(IO,*) 'YPLUSE'
            WRITE(IO,606) YPLUS
C
       WRITE(IO,656)LAST,L1,M1
656    FORMAT(/5X,'NO. ITERATIONS =',
      + I5,2X,'L1,M1 - ',2I3,/)
       WRITE(IO,657)(I,NTIMES(I),I=1,NFP3)
657    FORMAT(/2X,'EQN SWEEPS',/,6(5X,I2,1X,I2))
       IF (ITER.EQ.LAST) LSTOP=.TRUE.
C
C       WRITE DATA TO FILE 17 FOR USE BY PLOT PACKAGE
C
       WRITE(17,6800) L1,M1
       WRITE(17,6810) (X(I),I=1,L1)
       WRITE(17,6810) (Y(I),I=1,M1)
       WRITE(17,6810) (XU(I),I=1,L1)
       WRITE(17,6810) (YV(I),I=1,M1)
        RHO=RHOCON
        DO 4000 J = 1,M1
        WRITE(17,6820) (RHO,I=1,L1)
C       WRITE(IO,606) (CON(I,J),I=LFAB,L3)
 4000   CONTINUE
        DO 4010 J = 1,M1
        WRITE(17,6820) (U(I,J),I=1,L1)
 4010   CONTINUE
        DO 4020 J = 1,M1
        WRITE(17,6820) (V(I,J),I=1,L1)
 4020   CONTINUE
        DO 4030 J = 1,M1
        WRITE(17,6820) (P(I,J),I=1,L1)
 4030   CONTINUE
 6800    FORMAT(1X,2I5)
 6810    FORMAT(7(1X,E10.4))
 6820    FORMAT(6(1X,E12.6))
        RETURN
        END
C
```

```
      SUBROUTINE YYPLUS(CD25,RHO,AMU,DIST,TURB,YPLUS)
      IMPLICIT REAL*8 (A-H,O-Z)
606   FORMAT(7(1X,e9.3))
      YPLUS = RHO*CD25*SQRT(TURB)*DIST/AMU
      RETURN
      END
```

APPENDIX F
User Routine for Agent Transport

The subroutine USER to model the transport of agent is given here. This is for the case of radial injection as is the routine in Appendix D. The range of axial nodes for the injector are defined by LV1 and LV2 on page 126. The boundary condition is enforced following ENTRY BOUND on page 128 and the corresponding coefficients in the finite difference equations are given appropriate values following ENTRY GAMSOR on page 129. Note that the inlet condition as defined on page 128 has an exponential variation in time. The time part has no effect in the steady state solution because the exponential term is negligible. It comes into play only in the transient case and is used to mimic the inlet condition observed in experiments. In this case output is controlled directly in subroutine USER and not in a separate routine.

Output is given for the case given in Appendix D for the steady state situation.

A transient problem is not given here. The only difference is that NSTEADY would be set equal to 0, and for this case DT would be set equal to 1 (sec). In this case, of course, the exponential variation of concentration with time at the inlet, as defined at ENTRY BOUND, would come into play.

```
C
C.....................................................................
C
C            ------   T E S T    C H A M B E R   ------
C
C                            A G E N T
C
C.....................................................................
C
      SUBROUTINE USER
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
      PARAMETER (ID=151,JD=151,NFD=7,NFP3=7,MIJ=151,NREG=10)
C
      LOGICAL  LSOLVE,LBLK,LSTOP
C
      COMMON/BLOCK/F(ID,JD,NFD),GAM(ID,JD),CON(ID,JD),
     : AIP(ID,JD),AIM(ID,JD),AJP(ID,JD),AJM(ID,JD),AP(ID,JD),
     : X(ID),XU(ID),XDIF(ID),XCV(ID),XCVS(ID),
     : Y(JD),YV(JD),YDIF(JD),YCV(JD),YCVS(JD),
     : YCVR(JD),YCVRS(JD),ARX(JD),ARXJ(JD),ARXJP(JD),
     : R(JD),RMN(JD),SX(JD),SXMN(JD),XCVI(ID),XCVIP(ID)
C
      COMMON  DU(ID,JD),DV(ID,JD), FV(JD),FVP(JD),
     : FX(ID),FXM(ID),FY(JD),FYM(JD),PT(MIJ),QT(MIJ)
C
      COMMON/INDX/NF,NFMIN,NFMAX,NP,NRHO,NGAM,L1,L2,L3,M1,M2,M3,
     : IST,JST,ITER,LAST,IPREF,JPREF,MODE,NTIMES(NFP3)
C
      COMMON/LOGI/LSOLVE(NFP3),LBLK(NFP3)
C
      COMMON/VARI/TIME,DT,RELAX(NFP3),RHOCON
C
      COMMON/CNTL/LSTOP
C
      COMMON/SORC/SMAX,SSUM
C
      COMMON/COEF/FLOW,DIFF,ACOF
C
      COMMON/INOUT/IN,IO
C
      COMMON/PROP/ AMU,C1,C2,CD,CAPPA,ECON,PRTKE,PRTED,
     :             ALP,BET,ARAT,DIA,EPS,FABTHK,UINLET, UINJECT,
     :             CD5,CD25,CD75,C2M,TC2M,CDR2
C
      COMMON/BOX/ WF1,WF2,SCFAC,AA,BB,DISTN,DISTS,DISTE,DISTW,
     :            DISTNI,DISTEI,DISTWI
C
      COMMON/IBOX/ LB1,LB2,MB2,LNW1,LNW2,LSW1,LSW2,MNW,MSW,
     :             MEW1,MEW2,MWW1,MWW2,LEW,LWW,LF1,LF2,MF1,MF2,
     :             MNWI,LWWI,LEWI
C
      DIMENSION U(ID,JD),V(ID,JD),AGNT(ID,JD),AMUT(ID,JD)
C
      DIMENSION ICELL(NREG),IDCRS(NREG),JCELL(NREG),JDCRS(NREG),
     :          XL(NREG),YL(NREG),ENX(NREG),ENY(NREG)
C
      EQUIVALENCE (F(1,1,1),U(1,1)),(F(1,1,2),V(1,1)),
     :            (F(1,1,7),AGNT(1,1))
C
C         F(I,J,1) = U
C         F(I,J,2) = V
C         F(I,J,7) = C
C
C...PROBLEM DATA
C
```

```
      DATA IN,IO /15,16/
      DATA NFMIN,NFMAX /5,7/
      DATA NU,NV,NPC,NP,NTKE,NTED,NC /1,2,3,4,5,6,7/
C
      DATA LSOLVE /6*.false., .true./
C
      DATA LBLK  /7*.false./
C
      DATA NTIMES /7*1/
      DATA RELAX /7*1.0/
      DATA LSTOP /.FALSE./
      DATA TINY,BIG,SMALLEST,BIGGEST /1.E-30, 1.E+30, 1.E-70, 1.E+70/
      DATA ICOUNT / 0 /
C
C     -----------
      ENTRY INPUT
C     -----------
C
      READ(IN,*) NSTEADY
       DT = BIG
       IF (NSTEADY .NE. 1) READ(IN,*) DT
      READ(IN,*) LAST, NPRNT
      READ(IN,*) MODE, IPREF, JPREF
      READ(IN,*) RHOCON, AMU, ETA, ETAF, FABTHK, PRCON
C
      WRITE(IO,*) 'NSTEADY'
      WRITE(IO,6100) NSTEADY
      WRITE(IO,*) 'DT'
      WRITE(IO,6102) DT
      WRITE(IO,*) 'LAST, NPRNT, NTIMES(7)'
      WRITE(IO,6100) LAST, NPRNT, NTIMES(7)
      WRITE(IO,*) 'MODE, IPREF, JPREF'
      WRITE(IO,6100)  MODE, IPREF, JPREF
      WRITE(IO,*) 'RHOCON, AMU, ETA, ETAF, FABTHK, PRCON'
      WRITE(IO,6102) RHOCON, AMU, ETA, ETAF, FABTHK, PRCON
C
6100  FORMAT(15I5,/)
6102  FORMAT(7(1X,E10.4),/)
C
C     MESH GENERATION DATA
C
      read(in,*) ireg, (icell(i), idcrs(i), i=1,ireg)
      read(in,*) jreg, (jcell(j), jdcrs(j), j=1,jreg)
      read(in,*) (xl(i), enx(i), i=1,ireg)
      read(in,*) (yl(j), eny(j), j=1,jreg)
C
C     TEST CELL
C
      READ(IN,*) IBOX1,IBOX2,JBOX
C
C     ----------
      ENTRY GRID
C     ----------
C
C        L1       NUMBER OF U VELOCITY POINTS
C        M1       NUMBER OF V VELOCITY POINTS
C
C
      call meshgen(ireg,icell,idcrs,xl,enx,xu,l1)
      call meshgen(jreg,jcell,jdcrs,yl,eny,yv,m1)
C
      WRITE(IO,*) ' L1, M1'
      WRITE(IO,6100) L1,M1
      RETURN
C
C     -----------
```

```
      ENTRY START
C     -----------
C
C
C      READ RESTART FILE
C
       OPEN(UNIT=12,FILE='RESTART',FORM='UNFORMATTED')
C
C
      REWIND 12
      READ(12) LITER
      WRITE(IO,*) '  OLD ITER '
      WRITE(IO,6100) LITER
      DO 1  K = 1,6
       DO 1  J = 1,M1
        DO 1  I = 1,L1
         READ(12) F(I,J,K)
    1 CONTINUE
      write(io,*) 'READ F(I,J,K)  FROM RESTART FILE; LAST I,J,K '
      write(io,6100) i,j,k
      DO 2 J = 1,M1
        DO 2  I = 1,L1
         READ(12) AMUT(I,J)
    2 CONTINUE
      write(io,*) 'READ AMUT(I,J)  FROM RESTART FILE; LAST I,J'
      WRITE(IO,6100) I,J,K
C
C     GRID POINT RANGES FOR IMPERVIOUS BOX WALLS
C     AND THE RANGE FOR THE INJECTION VELOCITY
C
      LB1 = IBOX1 + 1
      LB2 = IBOX2 + 1
      MB1 = 1
      MB2 = JBOX + 1
C
C     RANGE FOR GRID POINTS NEXT TO N,S,E,W WALLS RESPECTIVELY
C
      LNW1 = 1
      LNW2 = L1
      MNW  = M2
      LSW1 = LB1
      LSW2 = LB2
      MSW  = MB2 + 1
      LEW  = LB1 - 1
      MEW1 = 1
      MEW2 = MB2
      LWW  = LB2 + 1
      MWW1 = 1
      MWW2 = MEW2
      MNWI = MB2 - 2
      LWWI = LB1 + 2
      LEWI = LB2 - 2
C
C     GRID POINT RANGES FOR FABRIC AND
C     THE WIDTH OF THE FABRIC CELLS
C
      LF1 = LB1
      LF2 = LB1 + 1
      MF1 = 1
      MF2 = MB2 - 1
      WF1 = XCV(LF1)
      WF2 = XCV(LF2)
C
C     GRID RANGES FOR INJECTION VELOCITY
C
      LV1 = IBOX1 + 5
      LV2 = IBOX2 - 2
```

```
C
C     SCALED COEFFICIENT FOR THE FABRIC
C
      SCFAC = FABTHK/(WF1 + WF2)
      ETAF = ETAF/SCFAC
C
C     OUTER RADIUS AND DISTANCES FROM WALLS TO
C     ADJACENT GRID POINT
C
      RMAX  = Y(M1)
      DISTN = YDIF(M1)
      DISTS = Y(MSW) - YV(MSW)
      DISTE = XU(LEW+1) - X(LEW)
      DISTW = X(LWW) - XU(LWW)
      DISTNI = Y(MNWI) - YV(MNWI)
      DISTEI = XU(LEWI+1) - X(LEWI)
      DISTWI = X(LWWI) - XU(LWWI)
C
C     WALL FACTORS
C
      WRITE(IO,*) '  AA, BB, SCFAC'
      WRITE(IO,6102) AA, BB, SCFAC
      WRITE(IO,*) '  RMAX, DISTN, DISTS, DISTE, DISTW'
      WRITE(IO,6102) RMAX, DISTN, DISTS, DISTE, DISTW
      WRITE(IO,*) '  DISTNI, DISTEI, DISTWI'
      WRITE(IO,6102) DISTNI, DISTEI, DISTWI
      WRITE(IO,*) '  IBOX1, IBOX2, JBOX'
      WRITE(IO,6100) IBOX1, IBOX2, JBOX
      WRITE(IO,*) '  LB1, LB2, MB1, MB2 '
      WRITE(IO,6100) LB1, LB2, MB1, MB2
      WRITE(IO,*) '  LNW1, LNW2, MNW, LSW1, LSW2, MSW'
      WRITE(IO,6100) LNW1, LNW2, MNW, LSW1, LSW2, MSW
      WRITE(IO,*) '  LEW, MEW1, MEW2, LWW, MWW1, MWW2'
      WRITE(IO,6100) LEW, MEW1, MEW2, LWW, MWW1, MWW2
      WRITE(IO,*) '  LEWI, LWWI, MNWI'
      WRITE(IO,6100) LEWI, LWWI, MNWI
      WRITE(IO,*) '  LF1, LF2, MF1, MF2'
      WRITE(IO,6100) LF1, LF2, MF1, MF2
      WRITE(IO,*) '  WF1, WF2'
      WRITE(IO,6102) WF1, WF2
C
C            ----- INITIAL CONDITIONS -----
C
      DO 11 J = 1,M2
       DO 11 I = 1,L1
        AGNT(I,J) = 0.0
   11 CONTINUE
       DO 12 J = 1,M1
        AGNT(1,J) = 1.0
   12 CONTINUE
      RETURN
C
C     -----------
      ENTRY DENSE
C     -----------
C
C
C     -----------
      ENTRY BOUND
C     -----------
C
C        AXIS AND OUTER WALL
C
      DO 300 I = 2,L1
         AGNT(I,1)  = AGNT(I,2)
         AGNT(I,M1) = AGNT(I,M2)
  300    CONTINUE
```

```
C
      DO 301 I = LV1,LV2
         AGNT(I,1) = 0.0
         AGNT(I,2) = 0.0
  301    CONTINUE
C
C
C        INLET AND OUTLET
C
      DO 302 J=1,M2
         AGNT(1,J)  = 1.0*(1.0 - exp(-TIME/100.))
         AGNT(L1,J) = AGNT(L2,J)
  302    CONTINUE
C
C        OUTER AND BACK WALLS OF CHAMBER
C
      DO 304 I = LB1,LB2
         AGNT(I,MB2+1) = AGNT(I,MB2+2)
  304    CONTINUE
C
      DO 306 J = 1,MB2
         AGNT(LB2+1,J) = AGNT(LB2+2,J)
  306    CONTINUE
C
C        INTERIOR WALLS
C
      DO 310 I = LWW1,LEWI
         AGNT(I,MNWI) = AGNT(I,MNWI-1)
  310    CONTINUE
C
      DO 312 J = 1,MNWI
         AGNT(LEWI,J) = AGNT(LEWI-1,J)
  312    CONTINUE
C
      RETURN
C
C     --------------
      ENTRY GRADIENT
C     --------------
      RETURN
C
C     ------------
      ENTRY GAMSOR
C     ------------
C
C         S = SC + SP*U  SO, AP = SP  &  CON = SC
C
      GO TO (499,499,499,499,499,499,400),NF
C
C               -------- C - CONCENTRATION --------
C
400   CONTINUE
C
      DO 405 J = 1,M1
      DO 405 I = 1,L1
         GAM(I,J) = RHOCON*ETA + AMUT(I,J)/PRCON
  405 CONTINUE
      DO 406 I = 1,L1
         GAM(I,1) = 0.0
  406 CONTINUE
      DO 407 J = 1,M1
         GAM(L1,J) = 0.0
  407 CONTINUE
C                        BOX WALLS
      DO 408 J = 1,MB2
         AP(LB2,J)    = -BIG
```

```
            GAM(LB2,J) = 0.0
            GAM(LB2+1,J) = 0.0
            GAM(LB2-1,J) = 0.0
  408       CONTINUE
        DO 409 I = LB1,LB2
            AP(I,MB2)    = -BIG
            GAM(I,MB2) = 0.0
            GAM(I,MB2+1) = 0.0
            GAM(I,MB2-1) = 0.0
  409       CONTINUE
C
C                                BOX
        DO 410 J = 1,MNWI
        DO 410 I = LWWI,LEWI
            GAM(I,J) = RHOCON*ETA
  410       CONTINUE
C                                FABRIC
        DO 411 J = 1,MF2
        DO 411 I = LF1,LF2
            GAM(I,J) = RHOCON*ETAF
  411       CONTINUE
C                                INJECTOR
        DO 412 J = 1,2
        DO 412 I = LV1,LV2
            CON(I,J) = 0.0
            AP(I,J) = -BIG
  412       CONTINUE
  499   RETURN
C
C       ----------
        ENTRY WALL
C       ----------
C
C         AJP = An,   AJM = As,   AIP = Ae,   AIM = Aw
C
        RETURN
C       ------------
        ENTRY OUTPUT
C       ------------
C
        ICOUNT = ICOUNT + 1
        IF (ITER.NE.LAST .AND. NPRNT.LE.1) RETURN
C
        IF (ITER .EQ. LAST) GOTO 500
        IF (ICOUNT .NE. NPRNT) RETURN
  500   WRITE(IO,612) TIME
  612   FORMAT(//5X,'TIME = ',E10.4,/)
        WRITE(IO,614)
  614   FORMAT(/5X,'EAST WALL  ',/)
        WRITE(IO,606) (AGNT(LEW,J),J=1,M1)
        WRITE(IO,616)
  616   FORMAT(/5X,'WEST WALL ',/)
        WRITE(IO,606) (AGNT(LWW+1,J),J=1,M1)
        WRITE(IO,618)
  618   FORMAT(/5X,'OUTLET ',/)
        WRITE(IO,606) (AGNT(L1,J),J=1,M1)
        WRITE(IO,620)
  620   FORMAT(/,5X,'I = LB2',/)
        WRITE(IO,606) (AGNT(LB2,J), J=1,M1)
        WRITE(IO,622)
  622   FORMAT(/,5X, 'FOR I=LEW,LWW+1   J=1,5,10',/)
        LWP = LWW + 1
           WRITE(IO,606) (AGNT(I,1), I=LEW,LWP)
           WRITE(IO,606) (AGNT(I,5), I=LEW,LWP)
           WRITE(IO,606) (AGNT(I,10), I=LEW,LWP)
C
```

```
      WRITE(IO,623)
623   FORMAT(/,5X,'FOR  I  = 54,64,70   J = 1,MB2',/)
         WRITE(IO,606) (AGNT(54,J), J=1,MB2)
         WRITE(IO,606) (AGNT(64,J), J=1,MB2)
         WRITE(IO,606) (AGNT(70,J), J=1,MB2)
C
      ICOUNT = 0
      IF (ITER .NE. LAST) RETURN
      WRITE(IO,656)LAST,L1,M1
656   FORMAT(/5X,'NO. ITERATIONS =',
     + I5,2X,'L1,M1 - ',2I3,/)
      WRITE(IO,657)(I,NTIMES(I),I=1,NFP3)
657   FORMAT(/2X,'EQN SWEEPS',/,6(5X,I2,1X,I2))
      IF (ITER.EQ.LAST) LSTOP=.TRUE.
C
C      WRITE DATA TO FILE 17 FOR USE BY PLOT PACKAGE
C
      WRITE(17,6800) L1,M1
      WRITE(17,6800) L1,M1
      WRITE(17,6810) (X(I),I=1,L1)
      WRITE(17,6810) (Y(I),I=1,M1)
       DO 4000 J = 1,M1
       WRITE(17,606) (AGNT(I,J),I=1,L1)
 4000  CONTINUE
 606   FORMAT(7(1X,e9.3))
 6800   FORMAT(1X,2I5)
 6810   FORMAT(7(1X,E10.4))
C
      RETURN
      END
c
      subroutine meshgen(nreg,ncell,ndcrs,el,en,x,nxs)
      implicit real*8 (a-h,o-z)
      dimension ncell(1),ndcrs(1),el(1),en(1),x(1)
c
      x(1) = 0.0
      x(2) = 0.0
      mo = 2
      mf = ncell(1) + mo
      nxs = mo
c
      do 1 n = 1,nreg
        mf = ncell(n) + mo
        x(mf) = x(mo) + el(n)
        pwr = 1.0/en(n)
        nx = ncell(n)
        nsgn = ndcrs(n)
        nxs = nxs + nx
        call coord(pwr,nx,nsgn,mo,mf,x)
        mo = mf
        mf = ncell(n+1) + mo
    1 continue
      return
      end
c
      subroutine coord(pwr,nx,nsgn,mo,mf,x)
      implicit real*8 (a-h,o-z)
      dimension x(1),si(100)
c
      dsi = 1.0/nx
      si(1) = 0.0
      si(nx+1) = 1.0
      do 1 i = 2,nx
        si(i) = si(i-1) + dsi
    1 continue
      do 2 i = 1,nx
```

```
      if (nsgn .ge. 0) then
        si(i+1) = si(i+1)**pwr
      else
        si(i+1) = 1.0 - (1.0 - si(i+1))**pwr
      endif
    2 continue
      xo = x(mo)
      xf = x(mf)
      nx1 = nx + 1
      do 3 i = 1,nx1
        ix = mo+i-1
        x(ix) = (xf - xo)*si(i) + xo
    3 continue
      return
      end
```

```
NSTEADY
    1
DT
 0.1000E+31
LAST, NPRNT, NTIMES(7)
 1000 1000    1
MODE, IPREF, JPREF
    2  120    2
RHOCON, AMU, ETA, ETAF, FABTHK, PRCON
 0.1062E-06 0.2750E-08 0.4400     0.1200E-01 0.2680E-01 0.7000E+00
 L1, M1
  122   32
 L1, M1
  122   32
              COMPUTATION FOR AXISYMMETRIC SITUATION
             ****************************************


  OLD ITER
 1000
READ F(I,J,K)  FROM RESTART FILE; LAST I,J,K
  123   33    7
READ AMUT(I,J)  FROM RESTART FILE; LAST I,J
  123   33    7
  AA, BB, SCFAC
 0.0000E+00 0.0000E+00 0.8933E-01
  RMAX, DISTN, DISTS, DISTE, DISTW
 0.4000E+01 0.3207E-01 0.3207E-01 0.1206E+00 0.1206E+00
  DISTNI, DISTEI, DISTWI
 0.4333E-01 0.7500E-01 0.7500E-01
  IBOX1, IBOX2, JBOX
   51   70   10
  LB1, LB2, MB1, MB2
   52   71    1   11
  LNW1, LNW2, MNW, LSW1, LSW2, MSW
    1  122   31   52   71   12
  LEW, MEW1, MEW2, LWW, MWW1, MWW2
   51    1   11   72    1   11
  LEWI, LWWI, MNWI
   69   54    9
  LF1, LF2, MF1, MF2
   52   53    1   10
  WF1, WF2
 0.1500E+00 0.1500E+00

     TIME = 0.9990E+33

     EAST WALL

 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.999E+00 0.990E+00 0.965E+00
 0.929E+00 0.929E+00 0.859E+00 0.884E+00 0.903E+00 0.924E+00 0.946E+00
 0.966E+00 0.981E+00 0.991E+00 0.996E+00 0.999E+00 0.100E+01 0.100E+01
 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
 0.100E+01 0.100E+01 0.100E+01 0.100E+01

     WEST WALL

 0.928E+00 0.928E+00 0.927E+00 0.927E+00 0.926E+00 0.926E+00 0.926E+00
 0.926E+00 0.927E+00 0.927E+00 0.923E+00 0.911E+00 0.922E+00 0.933E+00
 0.946E+00 0.960E+00 0.973E+00 0.985E+00 0.993E+00 0.997E+00 0.100E+01
 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
 0.100E+01 0.100E+01 0.100E+01 0.100E+01

     OUTLET
```

```
0.935E+00 0.935E+00 0.940E+00 0.946E+00 0.951E+00 0.957E+00 0.963E+00
0.968E+00 0.973E+00 0.978E+00 0.982E+00 0.985E+00 0.988E+00 0.990E+00
0.993E+00 0.995E+00 0.996E+00 0.998E+00 0.999E+00 0.999E+00 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01

    I = LB2

0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.919E+00 0.919E+00 0.929E+00
0.943E+00 0.958E+00 0.972E+00 0.983E+00 0.992E+00 0.997E+00 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01

    FOR I=LEW,LWW+1   J=1,5,10

0.100E+01 0.100E+01 0.999E+00 0.508E+00 0.276E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.181E-04 0.000E+00
0.928E+00 0.928E+00
0.999E+00 0.663E+00 0.523E+00 0.174E-01 0.114E-02 0.319E-03 0.355E-03
0.471E-03 0.410E-03 0.307E-03 0.193E-03 0.617E-04 0.663E-06 0.112E-05
0.156E-05 0.190E-05 0.244E-05 0.510E-05 0.510E-05 0.181E-04 0.000E+00
0.926E+00 0.926E+00
0.859E+00 0.353E-02 0.102E-02 0.133E-03 0.133E-03 0.133E-03 0.133E-03
0.133E-03 0.133E-03 0.125E-03 0.103E-03 0.624E-04 0.351E-04 0.279E-04
0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.000E+00
0.927E+00 0.927E+00

    FOR  I  = 54,64,70   J = 1,MB2

0.508E+00 0.508E+00 0.205E+00 0.754E-01 0.174E-01 0.166E-02 0.213E-03
0.182E-03 0.182E-03 0.133E-03 0.000E+00
0.000E+00 0.000E+00 0.668E-06 0.100E-05 0.112E-05 0.112E-05 0.129E-05
0.280E-05 0.280E-05 0.279E-04 0.000E+00
0.181E-04 0.181E-04 0.181E-04 0.181E-04 0.181E-04 0.182E-04 0.187E-04
0.198E-04 0.220E-04 0.257E-04 0.000E+00


    TIME = 0.1000E+34


    EAST WALL

0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.999E+00 0.990E+00 0.965E+00
0.929E+00 0.929E+00 0.859E+00 0.884E+00 0.903E+00 0.924E+00 0.946E+00
0.966E+00 0.981E+00 0.991E+00 0.996E+00 0.999E+00 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01

    WEST WALL

0.928E+00 0.928E+00 0.927E+00 0.927E+00 0.926E+00 0.926E+00 0.926E+00
0.926E+00 0.927E+00 0.927E+00 0.923E+00 0.911E+00 0.922E+00 0.933E+00
0.946E+00 0.960E+00 0.973E+00 0.985E+00 0.993E+00 0.997E+00 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01

    OUTLET

0.935E+00 0.935E+00 0.940E+00 0.946E+00 0.951E+00 0.957E+00 0.963E+00
0.968E+00 0.973E+00 0.978E+00 0.982E+00 0.985E+00 0.988E+00 0.990E+00
0.993E+00 0.995E+00 0.996E+00 0.998E+00 0.999E+00 0.999E+00 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01
```

```
    I = LB2

0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.919E+00 0.919E+00 0.929E+00
0.943E+00 0.958E+00 0.972E+00 0.983E+00 0.992E+00 0.997E+00 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01 0.100E+01
0.100E+01 0.100E+01 0.100E+01 0.100E+01

    FOR I=LEW,LWW+1   J=1,5,10

0.100E+01 0.100E+01 0.999E+00 0.508E+00 0.276E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.181E-04 0.000E+00
0.928E+00 0.928E+00
0.999E+00 0.663E+00 0.523E+00 0.174E-01 0.114E-02 0.319E-03 0.355E-03
0.471E-03 0.410E-03 0.307E-03 0.193E-03 0.617E-04 0.663E-06 0.112E-05
0.156E-05 0.190E-05 0.244E-05 0.510E-05 0.510E-05 0.181E-04 0.000E+00
0.926E+00 0.926E+00
0.859E+00 0.353E-02 0.102E-02 0.133E-03 0.133E-03 0.133E-03 0.133E-03
0.133E-03 0.133E-03 0.125E-03 0.103E-03 0.624E-04 0.351E-04 0.279E-04
0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.257E-04 0.000E+00
0.927E+00 0.927E+00

    FOR  I  = 54,64,70   J = 1,MB2

0.508E+00 0.508E+00 0.205E+00 0.754E-01 0.174E-01 0.166E-02 0.213E-03
0.182E-03 0.182E-03 0.133E-03 0.000E+00
0.000E+00 0.000E+00 0.668E-06 0.100E-05 0.112E-05 0.112E-05 0.129E-05
0.280E-05 0.280E-05 0.279E-04 0.000E+00
0.181E-04 0.181E-04 0.181E-04 0.181E-04 0.181E-04 0.182E-04 0.187E-04
0.198E-04 0.220E-04 0.257E-04 0.000E+00

    NO. ITERATIONS = 1000  L1,M1 - 122 32


 EQN SWEEPS
     1  1      2  1      3  1      4  1      5  1      6  1
     7  1
```

APPENDIX G
Vector Plotting Program

The vector plotting routine that follows was developed specifically for use with the code used to solve the flow field. It uses the plot file generated by the user routines for the majority of the input. The only other input required is prompted for at the beginning of a run. The prompts are self explanatory. Since the u and v velocity components are computed at different locations they are interpolated to the main grid points before the vectors for plotting are defined. This is accomplished in subroutine SPEED. The arrow heads for the vectors are constructed in subroutine ARROW. The code uses SUNCORE graphics routines which are no longer supported by SUN Microsystems. However, these routines are still available in "old" libraries. The code can be readily converted to another graphics system if required.

```
c
c     v e c t o r   p l o t s   u s i n g   s u n c o r e
c
c     input is either prompted from the screen and entered from the
c     keyboard or it is from the plot data output file from the
c     model.  UNIT 15 is used to read the plot datafile.  The name of
c     the plot datafile is prompted for.  Prompted data is self explanatory
c
      include "/usr/include/f77/usercore77.h"
c
      parameter (id = 500, jd = 500)
      integer vsurf(VWSURFSIZE)
      integer pixwindd
      external pixwindd
      integer InitializeCore, InitializeVwsurf, SelectVwsurf
c
      dimension x(id),y(jd),xg(id),yg(jd),u(id,jd),v(id,jd),
     :          ug(id,jd),vg(id,jd)
c
c     x & y are input grid points
c     xg & yg are shifted and scaled grid points
c     u & v are cell face velocities
c     ug & vg are grid point velocities
c
      character*6 pfile
      character*60  title
      character*60  subtitle
c
      data vsurf /VWSURFSIZE*0/
c
      write(*,60)
   60 format(2x,'enter input datafile name (usually PLOT15), format a6')
      read(*,61) pfile
   61 format(a6)
      write(*,62)
   62 format(2x,'enter title, format a60')
      read(*,64)  title
      write(*,63)
   63 format(2x,'enter subtitle, format a60')
      read(*,64) subtitle
   64 format(a60)
      write(*,66)
   66 format(2x,'enter NSYMX, NSYMY : = 0  nonsymmetry, = 1 symmetry')
      read(*,*) nsymx, nsymy
      write(*,68)
   68 format(2x,'enter reduction factor for velocities - e.g. 6.0')
      read(*,*) vscale
      write(*,69)
   69 format(/2x,'enter INCR --- plotting increment ',/,
     : 2x,'e.g. 1 plots vectors at every point, 2 at every other')
      read(*,*) incr
      write(*,70)
   70 format(/2x,'do you want subregion plotting ?',/,
     :       2x,'enter 1 for yes or 0 for no.  if you enter 1',/,
     :       2x,'on the next line enter the indicies for the region',/,
     :       2x,'to be plotted  IFIRST, ILAST, JFIRST, JLAST')
      read(*,*) imag
      if (imag.eq.1) read(*,*) ifst,ilst,jfst,jlst

c
c
c
      vsurf(DDINDEX) = loc(pixwindd)
      if (InitializeCore(BASIC, NOINPUT, TWOD) .ne. 0) call exit(1)
      if (InitializeVwsurf(vsurf, FALSE) .ne. 0) call exit(2)
      if (SelectVwsurf(vsurf) .ne. 0) call exit(3)
```

```
c
c
      open(unit=15,file=pfile,status='old')
c
c     since SUNCORE assumes a window y:x = 3:4
c     the program assumes a window from 0.0 to 16.0 in
c     the horizontal direction and from 0.0 to 12.0 in
c     the vertical direction.  Plotting takes place in
c     an 11.0 by 11.0 region starting at 4.0, 0.5.  The
c     remaining space can be used for textual information.
c
c     setup window coordinates
c
      xlft = 0.0
      xrgt = 16.0
      ybot = 0.0
      ytop = 12.0
      xo = 4.0
      yo = 0.5
      size = 11.0
c
      call input(id,jd,nx,ny,x,y,u,v,pfile)
c
      call SetNdcSpace2(1.0, 0.75)
c
c     use the following call to scale horiz and vertical directions
c     to compensate for the difference in the number of pixels in
c     each direction
c
      call SetViewport2(0.0,1.0, 0.0,0.75)
      call SetWindow(xlft, xrgt, ybot, ytop)
      call CreateTempSeg()
c
      call coords(nx,ny,x,y,xo,yo,nsymx,nsymy,size,xg,yg,
     :            imag,ifst,ilst,jfst,jlst)
      call speed(id,jd,nx,ny,u,v,ug,vg,vmax)
      call velplt(id,jd,xg,yg,ug,vg,xo,yo,nsymx,nsymy,size,vmax,
     :            vscale,incr,ifst,ilst,jfst,jlst)
c
c
        xt = 0.5
        yt = 11.75
        call label(xt,yt,title(1:60))
        yt = 11.55
        call label(xt,yt,subtitle(1:60))
c
c
c                 close and exit
c
      call CloseTempSeg()
      call sleep(60)
      call DeselectVwsurf(vsurf)
      call TerminateCore()
      end
c
      subroutine input(id,jd,nx,ny,x,y,u,v,pfile)
        dimension x(1),y(1),u(id,jd),v(id,jd)
        character*6 pfile
        open(unit=15,file=pfile,status='old')
c
        read(15,*) nx,ny
        read(15,*) (x(i), i=1,nx)
        read(15,*) (y(j), j=1,ny)
c
c         note that xu, yv & rho are not used but must be read
```

```
c
      read(15,*) (xu, i=1,nx)
      read(15,*) (yv, j=1,ny)
    do 1 j = 1,ny
      read(15,*) (rho, i = 1,nx)
  1   continue
    do 2 j = 1,ny
      read(15,*) (u(i,j), i = 1,nx)
  2   continue
    do 3 j = 1,ny
      read(15,*) (v(i,j), i = 1,nx)
  3 continue
c
      close(unit=15,status='keep')
    return
    end
c
    subroutine coords(nx,ny,x,y,xo,yo,nsymx,nsymy,size,xg,yg,
   :                  imag,ifs,ils,jfs,jls)
      dimension x(1),y(1),xg(1),yg(1)
c
c     xo & yo origin
c     plot window is size by size
c
      if (imag.eq.0) then
       ifs = 1
       ils = nx
       jfs = 1
       jls = ny
      endif
      sca = x(ils) - x(ifs)
      scay = y(jls) - y(jfs)
      if (nsymx.eq.1) sca = 2.0*sca
      if (nsymy.eq.1) scay = 2.0*scay
      if (sca.lt.scay) sca = scay
      if (nsymx.eq.1) xo = xo + size/2.0
      if (nsymy.eq.1) yo = yo + size/2.0
      do 10 i = 1,nx
         xg(i) = (x(i) - x(ifs))*size/sca + xo
 10   continue
      do 11 j = 1,ny
         yg(j) = (y(j) - y(jfs))*size/sca + yo
 11   continue
      return
      end
c
      subroutine speed(id,jd,nx,ny,u,v,ug,vg,vmax)
        dimension u(id,jd),v(id,jd),ug(id,jd),vg(id,jd)
c
c     finds velocity components at the grid points by interpolation
c     of cell face components.  the grid points lie half way
c     between the cell faces, only the interior points are actually
c     used
c
c     corner points
c
      ug(1,1) = u(2,1)
      vg(1,1) = v(1,2)
      ug(nx,1) = u(nx,1)
      vg(nx,1) = v(nx,2)
      ug(1,ny) = u(2,ny)
      vg(1,ny) = v(1,ny)
      ug(nx,ny) = u(nx,ny)
      vg(nx,ny) = v(nx,ny)
c
      n1 = nx - 1
```

```
      ml = ny - 1
c
c     bottom & top points
c
      do 2 i = 2,nl
      ug(i,1) = (u(i,1) + u(i+1,1))/2.0
      vg(i,1) = v(i,2)
      ug(i,ny) = (u(i,ny) + u(i+1,ny))/2.0
      vg(i,ny) = v(i,ny)
    2 continue
c
c     left & right points
c
      do 4 j = 2,ml
      ug(1,j) = u(2,j)
      vg(1,j) = (v(1,j) + v(1,j+1))/2.0
      ug(nx,j) = u(nx,j)
      vg(nx,j) = (v(nx,j) + v(nx,j+1))/2.0
    4 continue
c
c     interior points
c
      do 6 i = 2,nl
      do 6 j = 2,ml
      ug(i,j) = (u(i,j) + u(i+1,j))/2.0
      vg(i,j) = (v(i,j) + v(i,j+1))/2.0
    6 continue
c
c     max speed
c
      vmax = -1.0e30
      nxl = nx-1
      nyl = ny-1
      do 8 i =  2,nxl
      do 8 j =  2,nyl
        spd = sqrt(ug(i,j)**2 + vg(i,j)**2)
        if (vmax .lt. spd) vmax=spd
    8 continue
      return
      end
c
      subroutine velplt(id,jd,xg,yg,ug,vg,xo,yo,nsymx,nsymy,
     :                  size,vmax,vscale,incr,ifs,ils,jfs,jls)
        dimension xg(1),yg(1),ug(id,jd),vg(id,jd),xx(5),yy(5)
c
c     plot computation domain
c
      xx(1) = xg(ifs)
      xx(2) = xg(ils)
      if (nsymx.eq.1) xx(1) = 2.0*xo - xx(2)
      xx(3) = xx(2)
      xx(4) = xx(1)
      xx(5) = xx(1)
      yy(1) = yg(jfs)
      yy(4) = yg(jls)
      if (nsymy.eq.1) yy(1) = 2.0*yo - yy(4)
      yy(2) = yy(1)
      yy(3) = yy(4)
      yy(5) = yy(1)
      n5 = 5
c
      call moveabs2(xx(1),yy(1))
      call polylineabs2(xx,yy,n5)
c
c     plot frame
c
```

```
      xx(1) = xo
      if (nsymx.eq.1) xx(1) = xo - size/2.
      xx(2) = xx(1) + size
      xx(3) = xx(2)
      xx(4) = xx(1)
      xx(5) = xx(1)
      yy(1) = yo
      if (nsymy.eq.1) yy(1) = yo - size/2.
      yy(2) = yy(1)
      yy(3) = yy(1) + size
      yy(4) = yy(3)
      yy(5) = yy(1)
c
      call moveabs2(xx(1),yy(1))
      call polylineabs2(xx,yy,n5)
c
c     plot vectors
c
      nx1 = ifs + 1
      ny1 = jfs + 1
      nx2 = ils - 1
      ny2 = jls - 1
      do 10 i = nx1,nx2,incr
      do 10 j = ny1,ny2,incr
      xx(1) = xg(i)
      yy(1) = yg(j)
        xx(2) = xx(1) + ug(i,j)/(vmax*vscale)
        yy(2) = yy(1) + vg(i,j)/(vmax*vscale)
        call arrow (xx,yy,xo,yo,nsymx,nsymy,size)
   10 continue
      return
      end
c
      subroutine arrow(xx,yy,xo,yo,nsymx,nsymy,size)
      dimension xx(1),yy(1),xa(3),ya(3),xs(2),ys(2),
     :          xsa(3),ysa(3)
c
c                          yl *                    * xl
c                              *                *
c                                *            *
c                                  *        *
c                                    *    *
c                                      *
c                                    *   *
c                    s     *         * sy
c                        *           *
c                  *   angle         *
c              ************************
c                        sx
c
c     xl & yl local axes at the arrowhead
c     xa & ya are arrowhead coords in the xg yg system
c
c     xs, ys are used for symmetry plots of the vector
c     xsa, ysa are used for symmetry plots of the arrowhead
c     sx & sy are coords of tip relative to the tail
c     cs & sn are the cosine and sine of the angle
c
      pi8 = 3.14158/8.0
      sx = xx(2) - xx(1)
      sy = yy(2) - yy(1)
      s = sqrt(sx**2 + sy**2)
       if (s .lt. 10.e-10) return
      sa = s/5.
      xl = -sa*cos(pi8)
      yl = sa*sin(pi8)
      cs = sx/s
```

```
      sn = sy/s
c
c     arrowhead coordinates
c
      xa(1) = xx(2) + xl*cs - yl*sn
      ya(1) = yy(2) + xl*sn + yl*cs
      xa(2) = xx(2)
      ya(2) = yy(2)
      xa(3) = xx(2) + xl*cs + yl*sn
      ya(3) = yy(2) + xl*sn - yl*cs
c
c     draw arrow
c
      n2 = 2
      n3 = 3
      call moveabs2(xx(1),yy(1))
      call polylineabs2(xx,yy,n2)
      call moveabs2(xa(1),ya(1))
      call polylineabs2(xa,ya,n3)
c
      if (nsymx.eq.1) then
      xs(1) = 2.*xo - xx(1)
      xs(2) = 2.*xo - xx(2)
      xsa(1) = 2.*xo - xa(1)
      xsa(2)  = xs(2)
      xsa(3) = 2.*xo - xa(3)
c
c     draw arrow
c
      call moveabs2(xs(1),yy(1))
      call polylineabs2(xs,yy,n2)
      call moveabs2(xsa(1),ya(1))
      call polylineabs2(xsa,ya,n3)
c
      endif
      if (nsymy.eq.1) then
      ys(1) = 2.*yo - yy(1)
      ys(2) = 2.*yo - yy(2)
      ysa(1) = 2.*yo - ya(1)
      ysa(2) = ys(2)
      ysa(3) = 2.*yo - ya(3)
c
c     draw arrow
c
      call moveabs2(xx(1),ys(1))
      call polylineabs2(xx,ys,n2)
      call moveabs2(xa(1),ysa(1))
      call polylineabs2(xa,ysa,n3)
c
      endif
      if (nsymx.eq.1 .and. nsymy.eq.1) then
c
c     draw arrow
c
      call moveabs2(xs(1),ys(1))
      call polylineabs2(xs,ys,n2)
      call moveabs2(xsa(1),ysa(1))
      call polylineabs2(xsa,ysa,n3)
c
      endif
c
      return
      end
c
      subroutine label(x,y,sv)
      character *(*) sv
```

```
 call moveabs2(x,y)
 call text(sv)
return
end
```

APPENDIX H
Contour Plotting Program

The code used for plotting contours follows. The input for the code is explained on the first page of the code. The code plots contours on a cell by cell basis. A cell is defined by four grid points. The locations along the sides of a cell through which contours pass are found by interpolation of the values at the grid points. All the contours passing through a given cell are plotted before moving on to the next cell. The code was developed to use the grid numbering scheme used in the solution code. The solution code calculates all scalars such as pressure, concentration, temperature, etc. at the main grid points so these can be used directly by the contour plotting package. To plot streamlines, however, the velocity components must be interpolated from cell faces to the grid points. This is accomplished in subroutine STREAM. Integration of the velocity to compute the stream function is also performed here. The variable T(I,J) is used to store the values for contour plotting. As for the vector plotting code, this code is written using SUNCORE graphics routines but could be easily converted to some other system.

```
C
C          C O N T O U R    P L O T T E R
C
C     THREE FILES ARE OPENED FOR INPUT AND OUTPUT THEY ARE:
C          10       INPUT5      THIS CONTROLS THE PLOTTING
C          15       PLOT15      THIS IS THE PLOT FILE FROM THE
C                                FLOW CALCULATIONS
C          16       CNTRDAT     PRINTED OUTPUT
C
C      INPUT DATA IS FREE FIELD
C
C      DATA IN THE CODE PROMPTS YOU FOR AND THAT IS ENTERED
C      FROM THE KEYBOARD
C
C     NSETS, NSTRM
C                    WHERE      NSETS     NUMBER OF SETS OF SCALARS FOR
C                                         WHICH CONTOUR PLOTS ARE TO
C                                         TO BE MADE
C                               NSTRM     = 0  NO STREAMLINES
C                                         = 1  STREAMLINES
C
C     OTHER PROMPTED INPUT IS SELF EXPLANATORY
C
C     DATA FROM FILE INPUT5
C
C     NCNTRS, NTIMES, NSYMX, NSYMY, MODE
C     NBND
C           IF NBND = 1  THEN
C     (NPTS(J), J = 1,NBND)
C     (XB(I,J), YB(I,J), I = 1,K)
C
C        WHERE       NCNTRS      NUMBER OF CONTOURS TO BE PLOTTED
C                    NTIMES      0 OR 1 (NOT CURRENTLY USED)
C                    NSYMX       1  SYMMETRIC W.R.T. HORIZONTAL X-AXIS
C                                0  NOT SYMMETRIC
C                    NSYMY       1  SYMMETRIC W.R.T. VERTICAL Y-AXIS
C                                0  NOT SYMMETRIC
C                    MODE        1  CARTESIAN COORDINATES
C                                2  AXISYMMETRIC CYLINDRICAL COORDINATES
C
C                    NBND        NUMBER OF BOUNDARIES TOBE PLOTTED FOR
C                                HIGHLIGHTING THE PROBLEM DOMAIN
C                    NPTS        NUMBER OF POINTS ON EACH CONTINUOUS
C                                BOUNDARY
C                    XB,YB       X & Y COORDINATES OF THE CONTINUOUS
C                                SECTIONS OF THE BOUNDARIES
C
C
      include "/usr/old/usercore77.h"
c
      PARAMETER (ID=500,JD=500)
      DIMENSION P(ID),P5(ID),X(ID),Y(ID),NL(4),EL(4),T1(4),T2(4),
     1          XX(4),YY(4),XC(2),YC(2),DEL(ID,4),TP(ID,4),
     2          XS(ID,4),YS(ID,4),XB(21,60),YB(21,60),NPTS(21),
     3          XF(5),YF(5),XG(ID),YG(ID)
      COMMON T(ID,JD),R(ID,JD),U(ID,JD),V(ID,JD)
      CHARACTER*60 TITLE, SUBTITLE, SSTITLE1
      CHARACTER*10 NAME1, NAME2, NAME3
      CHARACTER*10 VAL1, VAL2, VAL3
C
C*************************************************************************
C
C          INITIATE PLOTTER
C
      integer vsurf(vwsurfsize)
      integer pixwindd
```

```
      external pixwindd
      integer initializecore, initializevwsurf, selectvwsurf
      data vsurf /vwsurfsize*0/
C
      DATA NAME1, NAME2, NAME3 /'MIN', 'MAX', 'NUMBER' /
      DATA SSTITLE1 /'STREAMLINES'/
C
      OPEN(UNIT=10,FILE='INPUT5',STATUS='OLD')
      OPEN(UNIT=15,FILE='PLOT15',STATUS='OLD')
      OPEN(UNIT=16,FILE='CNTRDAT',STATUS='OLD')
      OPEN(UNIT=25,FILE='CHARACT',STATUS='OLD')
C
      WRITE(*,660)
 660  FORMAT(/2X,'ENTER NSETS & NSTRM')
      READ(*,*) NSETS, NSTRM
      WRITE(*,667)
      READ(*,661) TITLE
 661  FORMAT(A60)
      WRITE(*,668)
      READ(*,661) SUBTITLE
 667  FORMAT(/2X,'ENTER TITLE')
 668  FORMAT(/2X,'ENTER SUBTITLE')
      WRITE(*,669)
 669  FORMAT(/2X,'DO YOU WANT TO PLOT A SUBREGION ?',/,
     :       2X,'ENTER 1 FOR YES OR 0 FOR NO.  IF YOU ENTER 1',/,
     :       2X,'ON THE NEXT LINE ENTER THE INDICIES FOR THE REGION',/,
     :       2X,'TO BE PLOTTED:  IFIRST,ILAST, JFIRST, JLAST')
      READ(*,*) IMAG
      IF (IMAG.EQ.1) READ(*,*) IFST,ILST,JFST,JLST
C
c     open suncore graphics
c
      vsurf(ddindex) = loc(pixwindd)
      if (initializecore(basic, noinput, twod) .ne. 0) call exit(1)
      if (initializevwsurf(vsurf, false) .ne. 0) call exit(2)
      if (selectvwsurf(vsurf) .ne. 0) call exit(3)
c
c     since SUNCORE assumes a window y:x = 3:4
c     the program assumes a window from 0.0 to 16.0 in
c     the horizontal direction and from 0.0 to 12.0 in
c     the vertical direction.  Plotting takes place in
c     an 11.0 by 11.0 region starting at 4.0, 0.5.  The
c     remaining space can be used for textual information.
c
c     setup window coordinates
c
      xlft = 0.0
      xrgt = 16.0
      ybot = 0.0
      ytop = 12.0
      xo = 4.0
      yo = 0.5
      size = 11.0
c
      call setndcspace2(1.0, 0.75)
      call setviewport2(0.0,1.0, 0.0,0.75)
      call setwindow(xlft, xrgt, ybot, ytop)
      call createtempseg()
c
      LBUG = 0
C
C**********************************************************************
C
C          INPUT PLOT CONTROL PARAMETERS USING FILE 10
C
      READ(10,*) NCNTRS,NTIMES,NSYMX,NSYMY,MODE
```

```
C
      MSET = 1
C
      READ(10,*) NBND
      IF (NBND.EQ.0) GO TO 1
      READ(10,*) (NPTS(J),J=1,NBND)
      DO 100 J = 1,NBND
      K = NPTS(J)
      READ(10,*) (XB(I,J), YB(I,J), I = 1,K)
 100  CONTINUE
 1    CONTINUE
C
C          INPUT GEOMETRY AND FIELD VALUES USING FILE 15
C
      READ(15,*) NX,NY
      READ(15,*) (X(I), I = 1,NX)
      READ(15,*) (Y(I), I = 1,NY)
      IF (NSTRM.EQ.1) READ(15,*) (XU, I = 1,NX)
      IF (NSTRM.EQ.1) READ(15,*) (YV, I = 1,NY)
C
      IF (LBUG.GT.0) THEN
       WRITE(16,*) 'NCNTRS,NVALU,NX,NY'
       WRITE(16,*) NCNTRS,NVALU,NX,NY
      ENDIF
c
      NSSTITLE = 0
      IF (NSTRM .EQ. 1) NSSTITLE = 1
1000   CALL STREAM(NX,NY,MODE,MSET,NSTRM,X,Y,PMAX,PMIN)
C
C          DEFINE THE SET OF CONTOURS
C
      PINC = (PMAX - PMIN)/(NCNTRS-1)
      PMIN = PMIN + PINC/1000.
      PMAX = PMAX - PINC/1000.
      PINC = (PMAX - PMIN)/(NCNTRS-1)
C
        REWIND 25
        WRITE(25,670) PMIN,PMAX,NCNTRS
 670    FORMAT(2(1X,E10.4),1X,I10)
        REWIND 25
        READ(25,671) VAL1, VAL2, VAL3
 671    FORMAT(3(1X,A10))
      IF (NCNTRS.EQ.1) PINC = 0.0
      TOL = PINC/10.
      P(1) = PMIN
      NLN = 1
      N5 = 0
      IF (NCNTRS.EQ.1) GO TO 132
      DO 130 I = 2,NCNTRS
      P(I) = P(I-1) + PINC
      IF (NLN .EQ. 5)  THEN
         N5 = N5 + 1
         P5(N5) = P(I)
      ENDIF
      NLN = NLN + 1
      IF (NLN .EQ. 6) NLN = 1
 130  CONTINUE
 132  CONTINUE
C
      IF (LBUG.EQ.1) WRITE(16,*) ' P',(P(I),I=1,NCNTRS)
C
c         plot frame
c
      xf(1) = xo
      xf(2) = xf(1) + size
      xf(3) = xf(2)
```

```
      xf(4) = xf(1)
      xf(5) = xf(1)
      yf(1) = yo
      yf(2) = yf(1)
      yf(3) = yf(1) + size
      yf(4) = yf(3)
      yf(5) = yf(1)
c
      line = 0
      call setlinestyle(line)
      call setlinewidth(0.2)
      call moveabs2(xf(1),yf(1))
      call polylineabs2(xf,yf,5)
      call setlinewidth(0.0)
c
c       label plot
c
      xt = 0.5
      yt = 11.75
      call label(xt,yt,title(1:60))
      yt = 11.5
      call label(xt,yt,subtitle(1:60))
      yt = 11.25
      if (nsstitle.eq.1) call label(xt,yt,sstitle1(1:60))
      nsstitle = 0
      yt = 10.0
      call label(xt,yt,name1(1:10))
      xt = 2.0
      call label(xt,yt,val1(1:10))
      xt = 0.5
      yt = 9.75
      call label(xt,yt,name2(1:10))
      xt = 2.0
      call label(xt,yt,val2(1:10))
      xt = 0.5
      yt = 9.5
      call label(xt,yt,name3(1:10))
      xt = 2.0
      call label(xt,yt,val3(1:10))
c
      call coords(nx,ny,x,y,nbnd,npts,xb,yb,xo,yo,nsymx,nsymy,
     :            xpo,ypo,size,xg,yg,imag,ifst,ilst,jfst,jlst)
C
      NSYM = 0
      IF (NTIMES.EQ.0) NTIMES = 1
ccccccccccc      DO 700 ITIME = 1,NTIMES
ccccccccccc      READ(15,*) ((T(I,J),I=1,NX),J=1,NY)
C
      IF (LBUG.EQ.1) THEN
      WRITE(16,*) '  NX,NY',NX,NY
      WRITE(16,*) '  I,J,T'
      DO 31 I=1,NX
      DO 31 J=1,NY
      WRITE(16,*) I,J,T(I,J)
   31 CONTINUE
      END IF
C
C          LOOP OVER ALL CELLS
C
      NC = ILST - 1
      NR = JLST - 1
   10 DO 600 LY = JFST,NR
      DO 600 LX = IFST,NC
C
      CALL WIDTH(LX,LY,NX,NY,XG,YG,T,EL,T1,T2,XX,YY)
C
```

```
      IF (LBUG .EQ. 1) THEN
      WRITE(16,*) '  LX,LY',LX,LY
      WRITE(16,*) '  T1',(T1(I),I=1,4)
      WRITE(16,*) '  T2',(T2(I),I=1,4)
      WRITE(16,*) '  XX',(XX(I),I=1,4)
      WRITE(16,*) '  YY',(YY(I),I=1,4)
      WRITE(16,*) '  EL',(EL(I),I=1,4)
      END IF
C
      K1 = 1
      K2 = 2
      DO 350 I = 1,4
C
      CALL POTEN(T1(I),T2(I),EL(I),P,PINC,NCNTRS,TP(1,I),DEL(1,I),JX)
C
      IF (LBUG.EQ.1) THEN
      WRITE(16,*) '  SIDE',I,'  TP'
      WRITE(16,*) (TP(JPX,I),JPX=1,NCNTRS)
      END IF
C
      NL(I) = JX
      IF (JX.EQ.0) GO TO 325
      DX = XX(K2) - XX(K1)
      DY = YY(K2) - YY(K1)
C
      IF (LBUG.EQ.1) THEN
      WRITE (16,*) ' K1 K2 DX DY',K1,K2,DX,DY
      END IF
C
      DO 300 J = 1,JX
      XS(J,I) = XX(K1) + DX*DEL(J,I)/EL(I)
      YS(J,I) = YY(K1) + DY*DEL(J,I)/EL(I)
 300  CONTINUE
C
      IF (LBUG.EQ.1) THEN
      WRITE(16,*) '  JX',JX
      WRITE(16,*) '  XS',(XS(J,I),J=1,JX)
      WRITE(16,*) '  YS',(YS(J,I),J=1,JX)
      END IF
C
 325  CONTINUE
C
      K1 = K1 + 1
      K2 = K2 + 1
      IF (K2.GT.4) K2 = 1
C
 350  CONTINUE
      IX = 2
      JX = NL(1)
 7    CONTINUE
      IF (JX.EQ.0) GO TO 8
      I1 = IX - 1
C
      DO 470 I = IX,4
      DO 450 J = 1,JX
      KX = NL(I)
      IF (IX.EQ.0) GO TO 450
      DO 400 K = 1,KX
      TST = ABS(TP(J,I1) - TP(K,I))
      IF(TST.GT.TOL) GO TO 400
      XC(1) = XS(J,I1)
      XC(2) = XS(K,I)
      YC(1) = YS(J,I1)
      YC(2) = YS(K,I)
C
      IF (NSYM.EQ.1) THEN
```

```
       IF (NSYMX.NE.0) THEN
        XC(1) = 2.*XPO - XC(1)
        XC(2) = 2.*XPO - XC(2)
        ENDIF
       IF (NSYMY.NE.0) THEN
        YC(1) = 2.*YPO - YC(1)
        YC(2) = 2.*YPO - YC(2)
        ENDIF
      ENDIF
C
C        LINE = 0  SOLID
C               1  DOT
C               2  DASH
C               3  DOT-DASH
C
      LINE = 1
      DO 360 I5 = 1,N5
      IF (ABS(P5(I5)-TP(K,I)) .LT. TOL) LINE = 0
 360  CONTINUE
      IF (ABS(P(1)-TP(K,I)) .LT. TOL) LINE = 2
      IF (ABS(P(NCNTRS)-TP(K,I)) .LT. TOL) LINE = 0
      call setlinestyle(line)
      call moveabs2(xc(1),yc(1))
      call polylineabs2(xc,yc,2)
 400  CONTINUE
 450  CONTINUE
 470  CONTINUE
 8    CONTINUE
      JX = NL(IX)
      IX = IX + 1
      IF (IX.LE.4)  GO TO 7
 600  CONTINUE
      NSYM = NSYM + 1
      IF(NSYM.LT.2 .AND. (NSYMX.NE.0 .OR. NSYMY.NE.0)) GO TO 10
C
 700  CONTINUE
c
c     hold plot for 60 seconds then clear screen
c
      call sleep(60)
      call newframe
c
      MSET = MSET + 1
C
      IF (MSET.LE.NSETS) GO TO 1000
C
      CLOSE(UNIT=10,STATUS='KEEP')
      CLOSE(UNIT=15,STATUS='KEEP')
      CLOSE(UNIT=16,STATUS='KEEP')
      CLOSE(UNIT=25,STATUS='KEEP')
C
C**************TERMINATE PLOTTER***********
C
c
c                   close and exit
c
      call CloseTempSeg()
      call DeselectVwsurf(vsurf)
      call TerminateCore()
c
      END
C
C*****************************************************************************
C                S U B R O U T I N E S
C*****************************************************************************
C
```

```
      SUBROUTINE WIDTH(I,J,NX,NY,X,Y,T,EL,T1,T2,XX,YY)
        PARAMETER (ID=500,JD=500)
      DIMENSION X(NX),Y(NY),T(ID,JD),EL(4),T1(4),T2(4),XX(4),YY(4)
C
C
C                              (3)
C                    4 O-------------------O 3
C                      I                   I
C                      I                   I
C                      I                   I
C                      I                   I
C                  (4) I                   I (2)
C                      I                   I
C                      I                   I
C                      I                   I
C                      I                   I
C                    1 O-------------------O 2
C                              (1)
C
C          LENGTHS OF THE SIDES
C
      EL(1) = ABS(X(I+1) - X(I))
      EL(3) = EL(1)
      EL(2) = ABS(Y(J+1) - Y(J))
      EL(4) = EL(2)
C
C          THE END VALUES FOR EACH SIDE
C
      T1(1) = T(I,J)
      T2(1) = T(I+1,J)
      T1(2) = T2(1)
      T2(2) = T(I+1,J+1)
      T1(3) = T2(2)
      T2(3) = T(I,J+1)
      T1(4) = T2(3)
      T2(4) = T1(1)
C
C          THE END POINT COORDS FOR EACH SIDE
C
      XX(1) = X(I)
      XX(2) = X(I+1)
      XX(3) = XX(2)
      XX(4) = XX(1)
      YY(1) = Y(J)
      YY(2) = YY(1)
      YY(3) = Y(J+1)
      YY(4) = YY(3)
C
      RETURN
      END
C
      SUBROUTINE POTEN(T1,T2,EL,P,PINC,NP,T,X,NX)
      DIMENSION P(1),T(1),X(1)
C
C     COMPUTE RELATIVE LOCATIONS OF CONTOURS ALONG A SIDE
C
      J = 1
      DT = T2 - T1
C
      IF (T1.EQ.T2) GO TO 4
      IF (T1.GT.T2) GO TO 1
      A = T1
      Z = T2
      GO TO 2
    1 A = T2
      Z = T1
```

```
    2  DO 100 I = 1,NP
       IF (A.LT.P(I)) GO TO 3
  100  CONTINUE
C
       GO TO 6
    3  A = P(I)
       IF (A.GT.Z) GO TO 6
       X(J) = EL*(A-T1)/DT
       T(J) = A
       J = J + 1
       A = A + PINC/10.
       GO TO 2
C
    4  DO 200 I = 1,NP
       IF (T1.EQ.P(I)) GO TO 5
  200  CONTINUE
C
       GO TO 6
    5  X(J) = 0.0
       T(J) = P(I)
       J = J + 1
C
C          NX = NUM OF CONTOURS PASSING THROUGH A SIDE
C
    6  NX = J - 1
C
       RETURN
       END
C
       SUBROUTINE STREAM(NX,NY,MODE,MSET,NSTRM,X,Y,TMAX,TMIN)
         PARAMETER (ID=500,JD=500)
       DIMENSION X(NX),Y(NY),UG(ID,JD),VG(ID,JD)
       COMMON T(ID,JD),R(ID,JD),U(ID,JD),V(ID,JD)
C
       IF (NSTRM.NE.1)  GOTO 100
        NSTRM = 0
C
         DO 40 J = 1,NY
         READ(15,*) (R(I,J),I=1,NX)
   40 CONTINUE
         DO 41 J = 1,NY
         READ(15,*) (U(I,J),I=1,NX)
   41 CONTINUE
         DO 42 J = 1,NY
         READ(15,*) (V(I,J),I=1,NX)
   42 CONTINUE
C
C      FIND VELOCITIES AT CELL GRID POINTS
C
       L = NX
       M = NY
C
C      CORNER POINTS
C
       UG(1,1) = U(2,1)
       VG(1,1) = V(1,2)
       UG(L,1) = U(L,1)
       VG(L,1) = V(L,2)
       UG(1,M) = U(2,M)
       VG(1,M) = V(1,M)
       UG(L,M) = U(L,M)
       VG(L,M) = V(L,M)
C
C      BOTTOM & TOP POINTS
C
      L1 = L-1
```

```
      DO 44 I = 2,L1
      UG(I,1) = (U(I,1) + U(I+1,1))/2.0
      VG(I,1) = V(I,2)
      UG(I,M) = (U(I,M) + U(I+1,M))/2.0
      VG(I,M) = V(I,M)
   44 CONTINUE
C
C     LEFT & RIGHT POINTS
C
      M1 = M-1
      DO 46 J = 2,M1
      UG(1,J) = U(2,J)
      VG(1,J) = (V(1,J) + V(1,J+1))/2.0
      UG(L,J) = U(L,J)
      VG(L,J) = (V(L,J) + V(L,J+1))/2.0
   46 CONTINUE
C
C     INTERIOR POINTS
C
      DO 48 I = 2,L1
      DO 48 J = 2,M1
      UG(I,J) = (U(I,J) + U(I+1,J))/2.0
      VG(I,J) = (V(I,J) + V(I,J+1))/2.0
   48 CONTINUE
C
      DO 50 J = 1,NY
      DO 50 I = 1,NX
      T(I,J) = 0.0
   50 CONTINUE
C
C     AVG. RHO*U AND RHO*V ON INTEGRATION PATH
C
C     LEFT BOUNDARY
C
      DO 60 J = 2,NY
      RU = (UG(1,J-1)*R(1,J-1) + UG(1,J)*R(1,J))/2.0
      IF (MODE.EQ.1) T(1,J) = T(1,J-1) + RU*(Y(J) - Y(J-1))
      IF (MODE.EQ.2) T(1,J) = T(1,J-1) -
     1                        RU*(Y(J)**2 - Y(J-1)**2)/2.0
   60 CONTINUE
C
C     BOTTOM BOUNDARY
C
      DO 62 I = 2,NX
      RV = (VG(I-1,1)*R(I-1,1) + VG(I,1)*R(I,1))/2.0
      IF (MODE.EQ.1) T(I,1) = T(I-1,1) - RV*(X(I) - X(I-1))
      IF (MODE.EQ.2) T(I,1) = 0.0
   62 CONTINUE
C
C     REMAINING POINTS
C
      DO 64 J = 2,NY
      DO 64 I = 2,NX
      RU = (UG(I,J)*R(I,J) + UG(I,J-1)*R(I,J-1))/2.0
c
c       remove ff stmt so only y  integration is performed
c
c       RV = (VG(I-1,J-1)*R(I-1,J-1) + VG(I,J-1)*R(I,J-1))/2.0
C
        IF (MODE.EQ.1) T(I,J) = T(I,J-1) + RU*(Y(J)-Y(J-1))
c
c       remove continuation of above stmt  so only y integ. is done
c
c     :                         - RV*(X(I)-X(I-1))
C
        IF (MODE.EQ.2) T(I,J) = T(I,J-1) - RU*(Y(J)**2-Y(J-1)**2)/2.0
```

```
C
   64  CONTINUE
C
       GOTO 110
C
  100  DO 70 J = 1,NY
        READ(15,*) (T(I,J),I=1,NX)
   70  CONTINUE
C
  110 CONTINUE
      TMAX = -1.E50
      TMIN =  1.E50
       DO 80 J = 1,NY
       DO 80 I = 1,NX
        IF(TMAX.LE.T(I,J)) TMAX = T(I,J)
        IF(TMIN.GE.T(I,J)) TMIN = T(I,J)
   80  CONTINUE
C
      RETURN
      END
c
      subroutine coords(nx,ny,x,y,nb,np,xb,yb,xo,yo,nsymx,nsymy,
     :                  xpo,ypo,size,xg,yg,imag,ifs,ils,jfs,jls)
        dimension x(1),y(1),xg(1),yg(1),np(1),xb(21,60),yb(21,60),
     :            xbd(21),ybd(21)
c
c     xo & yo origin
c     plot window is size by size
c
      if (imag.eq.0) then
       ifs = 1
       ils = nx
       jfs = 1
       jls = ny
      endif
      sca = x(ils) - x(ifs)
      scay = y(jls) - y(jfs)
      xpo = xo
      ypo = yo
      xbo = x(ifs)
      ybo = y(jfs)
      if (nsymx.eq.1) sca = 2.0*sca
      if (nsymy.eq.1) scay = 2.0*scay
      if (sca.lt.scay) sca = scay
      if (nsymx.eq.1) xpo = xo + size/2.0
      if (nsymy.eq.1) ypo = yo + size/2.0
      do 10 i = 1,nx
         xg(i) = (x(i) - x(ifs))*size/sca + xpo
   10 continue
      do 11 j = 1,ny
         yg(j) = (y(j) - y(jfs))*size/sca + ypo
   11 continue
      do 14 j = 1,nb
         ip = np(j)
         nsym = 0
   12 do 13 i = 1,ip
         xbd(i) = (xb(i,j) - xbo)*size/sca + xpo
         ybd(i) = (yb(i,j) - ybo)*size/sca + ypo
       IF (NSYM.EQ.1) THEN
       IF (NSYMX.NE.0) THEN
        XBD(I) = 2.*XPO - XBD(I)
        ENDIF
       IF (NSYMY.NE.0) THEN
        YBD(I) = 2.*YPO - YBD(I)
        ENDIF
      ENDIF
```

```
   13 CONTINUE
      nsym = nsym + 1
         call moveabs2(xbd(1),ybd(1))
         call polylineabs2(xbd,ybd,ip)
      if(nsym.lt.2 .and. (nsymx.ne.0 .or. nsymy.ne.0)) go to 12
   14 continue
      return
      end
c
      subroutine label(x,y,sv)
      character *(*) sv
       call moveabs2(x,y)
       call text(sv)
      return
      end
```